Received: 13 December 2019 | Revised: 12 May 2020 | Accepted: 25 May 2020

DOI: 10.1002/smr.2296

REVIEW ARTICLE

Journal of Software: Evolution and Process WILEY

Metrics in automotive software development: A systematic literature review

Martin Vogel[1] | Peter Knapik[2] | Moritz Cohrs[2] | Bernd Szyperrek[2] | Winfried Pueschel[2] | Haiko Etzel[2] | Daniel Fiebig[1] | Andreas Rausch[1] | Marco Kuhrmann[3]

[1]Institute for Applied Software Systems Engineering, Clausthal University of Technology, Goslar, Germany

[2]Volkswagen AG, Wolfsburg, Germany

[3]Faculty of Computer Science and Mathematics, University of Passau, Passau, Germany

Correspondence
Marco Kuhrmann, Faculty of Computer Science and Mathematics, University of Passau, Passau, Germany.
Email: kuhrmann@acm.org

Abstract

Software is an integrated part of new features within the automotive sector, car manufacturers, the Hersteller Initiative Software (HIS) consortium defined metrics to determine software quality. Yet, problems with assigning metrics to quality attributes often occur in practice. The specified boundary values lead to discussions between contractors and clients as different standards and metric sets are used. This paper studies metrics used in the automotive sector and the quality attributes they address. The HIS, ISO/IEC 25010:2011, and ISO/IEC 26262:2018 are utilized to draw a big picture illustrating (i) which metrics and boundary values are reported in literature, (ii) how the metrics match the standards, (iii) which quality attributes are addressed, and (iv) how the metrics are supported by tools. Our findings from analyzing 38 papers include a catalog of 112 metrics of which 17 define boundary values and 48 are supported by tools. Most of the metrics are concerned with source code, are generic, and not specifically designed for automotive software development. We conclude that many metrics exist, but a clear definition of the metrics' context, notably regarding the construction of flexible and efficient measurement suites, is missing.

KEYWORDS

automotive software development, quality attributes, software metrics, systematic literature review

1 | INTRODUCTION

Software has become key in the automotive industry and is an important driver for innovation, which is shown by approximately 80% of in-car functions that are realized through software.[1] It has become the norm that different software components communicate with each other within a car, across cars, and between cars and the environment. So-called *in-car functions*, such as the adaptive cruise control, as well as *cross-vehicle functions*, such as cooperative driver assistance systems or autonomous driving, are mainly realized through software.[1] However, the innovation and production life cycles of "classic" engineering components and software differ. Whereas the development of a new car takes on average 4 years, and it takes on average 7 years[1] until the launch, software evolves much faster.[1,2] That is, whereas the overall design of a new car matures relatively slow, software, like infotainment features, evolves much faster, thus resulting in a growing number of requirements and customer

expectations. Having an appropriate software design, it is also possible to deploy new functionality (e.g., over-the-air or online remote updates) after launch—even after features have been deployed. Because many vehicle functions are realized through software, quality of automotive software is key to participate in ensuring the reliability of a car.

Automotive software development is also challenged by the way software is developed today. Many software components are no longer developed in-house but outsourced to third parties.[1,3] Hence, car manufacturers have partly established procedures to efficiently and effectively assess such software components in the development process. Standards and norms like the ISO/IEC 26262:2018[4] and the ISO/IEC 25010:2011[5] provide support by defining development and quality assurance procedures. However, such standards are generic, thus requiring an adaptation and embodiment to the internal software development process. That is, generic quality requirements must be adapted to the actual software system measurement procedures, metrics must be mapped to quality attributes, and the boundary values have to be defined to fulfill these quality attributes. The characterization of software properties and the measurement of software quality can be done with the help of (software) metrics.[6,7] A multitude of metrics is available covering processes or products as a whole.[4,8-11] Many metrics are generic to allow for broad applicability and, to a certain extent, to support comparability of software components and systems. Metrics are available for different programming languages, models, and development activities, and a number of tools provide support for data collection and analysis.

1.1 | Problem statement and objective

Different definitions and a different understanding of quality attributes and metrics to assess relevant quality attributes lead to situations in which project parties (manufacturers, suppliers, and contractors) risk misunderstandings and costly renegotiations of qualities. An agreed, modern, and harmonized metric suite for automotive software development is not available, and moreover, a recommendation system that helps practitioners define the qualities of interest and to select metrics aligned with the available development tool chain is missing.

Our overall objective is to capture the current state of the art and practice of metrics and their use in automotive software development to support the improvement of quality management systems. We aim to provide views on metrics and quality standards that help consolidate metrics applied to automotive software projects and, thus, to reduce the necessity of deviations from defined qualities.

1.2 | Contribution

In this article, we present a systematic literature review on the use of software metrics in the automotive software sector. On the basis of 38 selected primary studies, we extracted 112 metrics for which we provide a detailed description and a categorization using the HIS* metric catalog, 31 selected quality attributes of the ISO/IEC 25010:2011[5] quality model, and the ISO/IEC 26262:2018 Part 6[4] as references. In the systematic review, we found only 17 metrics defining boundary values, yet, no general boundary values or recommendations were found. The metrics obtained from the systematic review were also analyzed in the context of 20 selected tools used to collect data and compute metrics. From the 112 metrics, 48 are supported by the selected tools. Our findings include that there is neither an agreement in literature on the practical relevance of specific metrics nor an agreed mapping between metrics and quality categories. Furthermore, our findings show that the metrics found are not specific for automotive software development. Metrics are of general nature and are used to assess software (as part of a safety-critical system) in general. Specific characterizations of metrics for automotive software development, notably a precise definition of the context of use, are not available. We contribute views proposing such mappings and characterizations that include the 112 metrics from the systematic review and the three selected standards as well as a mapping to 20 selected tools supporting these metrics. The mappings aim to help practitioners select appropriate metrics that cover desired quality attributes and that reduce the number of exceptions from the agreed quality requirements. Our results lay the foundation for constructing a recommendation system that helps practitioners develop adaptable measurement systems.

1.3 | Outline

The remainder of this article is organized as follows: Section 2 discusses background and related work. Section 3 presents the research design including the research goals, research questions, and the different steps performed to conduct our study. Section 4 presents the study results, which are discussed in Section 5, before we conclude the paper in Section 6. The appendix of this article includes detailed information about the study, detailed data tables of the study results, visual mappings, and the metric catalog.

*HIS stands for the "Herstellerinitiative Software" (OEM Software Initiative), a consortium composed of the German car manufacturers targeting automotive software development. This initiative, however, was closed in 2008. Information is partially available here: https://www.autosar.organd here: https://emenda.com/wp-content/uploads/2017/07/HIS-sc-metriken.1.3.1_e.pdf.

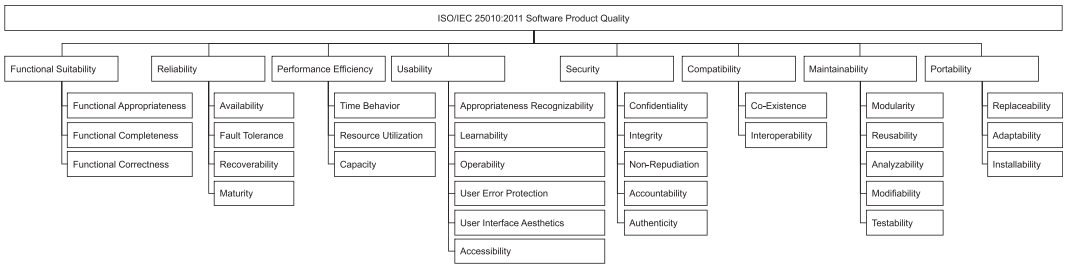


FIGURE 1 Overview of the software product quality attributes as defined by the ISO/IEC 25010:2011 standard, which have been selected for the analysis (see Figure 4)

2 | BACKGROUND AND RELATED WORK

We introduce the background regarding metrics and measurement systems in automotive software development before discussing related work.

2.1 | Metrics in automotive software development

In automotive software development, a number of quality standards, notably the ISO/IEC 26262:2018[4] (including the *Automotive Safety Integrity Levels*, ASIL) have to be applied. Quality norms and standards also refer to metrics, which are supposed to help assess software quality (e.g., Table C3). However, a clear understanding of which metrics in software development address a specific quality attribute (e.g., standard attributes as shown in Figure 1) is often missing. It is often unclear what a reasonable set of metrics addressing a specific quality attribute looks like. For instance, depending on different ASIL levels, metrics can be interpreted differently with regards to value ranges and boundary values. As a result, automotive software projects often struggle with inappropriate metrics, which need to be adapted, and deviations from initially defined target values have to be carefully documented.[†] This situation challenges company- and/or product-wide quality management systems in establishing standards and providing a notion of what is considered a high-quality software.

In response to the situation outlined above, in automotive software development, a standardized set of metrics was developed by the HIS consortium. The HIS metrics target the C programming language and provide no further categorization, for example, for project size, project type, and safety level. A definition of boundary values to be applied to the different project types is not provided. HIS does not include mappings of metrics to quality attributes of interest. Furthermore, as other languages than C are also used in automotive software development today, for example, C++ and Java, the applicability of the HIS metrics is limited. In practice, this situation leads to constant negotiations of boundary values and quality numbers that car manufacturers, integrators, and (external) software vendors have to agree upon. Because there is no standard set of metrics available for automotive software development, software quality is hard to compare across different projects and even within projects that include different software components.

Another perspective on software quality is given by the ISO/IEC 25010:2011.[5] This standard, among other things, provides a holistic perspective on general product quality. ISO/IEC 25010:2011 evolves the quality attributes from the ISO/IEC 9126 to allow for a better characterization of the different qualities to be considered in software and system development projects. Figure 1 provides an overview of the quality criteria, notably of those criteria that have been selected for analysis in the article at hand (Section 3.1). For all quality attributes, the complementing ISO/IEC 25023:2016[9] recommends metrics (note the standard also uses the term “measure”), including simple formulae and basic boundary values as baselines. The quality criteria illustrated in Figure 1 also show that the ISO/IEC 25010:2011 is meant to be applicable to any software-intensive system. Hence, in this article, these general quality criteria serve as a baseline for the literature analysis.

As the different standards for automotive software (ISO/IEC 26262:2018) and general software product quality (ISO/IEC 25010:2011) evolve, automotive software companies are nowadays challenged by different, in parts competing and inconsistent, metric sets, the one provided by the ISO/IEC 26262:2018 and the one provided by the ISO/IEC 25023:2016. Hence, for each automotive software project, these different metric sets have to be analyzed and implemented in projects carefully to avoid inconsistencies and the risk of misunderstanding due to deviating quality baselines. The article at hand specifically addresses this issue by providing mappings using the ISO/IEC 25010:2011 quality attributes as a reference to integrate and map the different standards and the metrics found in the systematic review.

[†]A car as such is a dependable system, i.e., reliability is a key requirement of the system. Consequently, the software of a car must also fulfill the quality attributes related to reliability, e.g., availability, safety, and security for which *dependability* is the umbrella (see also Avizienis et al.[27]).

2.2 | Related work

Metrics are an important topic in software engineering research,[6,8] and, therefore, metrics are constantly reported in literature. For instance, El-Sharkawy et al.[10] study the current state of practice regarding deployment metrics in *software product lines* (SPL). They point out that there are very few metrics that can assess quality in the context of SPLs. It was further pointed out that many metric definitions are rather inaccurate and reuse of metrics across (sub-)products has barely taken place. Furthermore, authors found that the information from the SPLs combined with the code artifacts received little attention. Authors conclude that it would be valuable to combine the information from the variability model and the code artifacts. In a similar direction, Wagner et al.[12] propose the *Quamoco* approach, which is a meta-quality model aiming at closing the gap between abstract quality attributes and concrete quality assessments. There exists a number of quality models,[13] and a number of them are (partially) implemented in tools that support collection of metric data and analyzing quality attributes of interest. For instance, Tsuda et al.[14] provide a measurement and quality evaluation framework based on the SQuaRE model, that is, the ISO/IEC 25000 standard series. However, most quality models are focused on specific aspects leaving out the big picture of the "product." Quamoco's base model is based on the ISO/IEC 25010 quality attributes (Figure 1) and includes more than 300 factors and 500 measures for software products developed in Java and C#.[15]

Besides secondary studies, quality models, and meta-quality models providing a big picture, several publications deal with the actual application of metrics to practice. For instance, Selvarani et al.[16] developed a model to examine and evaluate the *Chidamber and Kemerer* (CK) metrics for their predictive capability for errors and degeneration. The model was developed based on the "Shannon entropy." The result shows that the *NASA/Rosenberg threshold*[16] risk categorization allows for a high level of forecasting. Also in the Space domain, the European Cooperation for Space Standardization (ECSS) standards[17] provide guidelines and examples of software metrics. These metrics can be used in space system development with respect to the requirements[18] and to provide a coherent view of the software metrication program definition and implementation. In this article, we also use the ECSS standard to fill gaps in the details of metrics obtained from the systematic review (see also Section 3.3.2).

An analysis of complexity measures in practice was provided by Antinyan et al.[19] Authors found that the current measures for code quality and complexity measures are barely known in practice and rarely used by software engineers; also supported by Sloss et al.[11] Antinyan et al.[19] showed that the lack of knowledge has a negative impact on the internal quality of a software. Furthermore, they analyzed how software metrics integrate with the goals of an organization and conclude that it is important to capture well-designed metrics with documented goals in a catalog. This helps to measure progress and achieve the defined goals. A practical and systematic start-to-finish method for selecting, designing, and implementing metrics is considered a valuable aid in further improving software products, processes, and services. Such catalogs, however, require metrics to be "universal." In this regard, Hoffman et al.[20] investigated how metrics can be used universally. For this, authors analyzed the properties of metrics in detail and proposed a schema to assess metrics for their universal applicability. At the other end of the spectrum, Alves et al.[21] present a method that can be used to create boundary values for software metrics using benchmarks. Authors propose a method that weights software metrics and evaluated their proposal using 100 software projects. They conclude that their method can better reflect the boundary values of metrics, because essential metric properties have been taken into account. Similarly, Schroeder et al.[22] studied how software metrics combined with expert knowledge can be used to evaluate models for specific quality criteria. They analyzed 65 000 software revisions and showed that the predictive quality of models can be improved using expert knowledge and software metrics together. Finally, Schroeder et al.[23] studied how machine learning methods and software metrics can predict the development of models (Matlab/Simulink) in the automotive industry. Authors analyzed a project with in total 4547 revisions. They could show that metrics provide an important input to support the machine learning methods.

The article at hand contributes an analysis of metrics as reported to be used in automotive software development. We aim at developing a catalog of metrics for this domain to lay the foundation to develop recommendation systems that improve the usability of metric-based measurement systems in industrial contexts. Furthermore, we aim at consolidating the variety of metrics available to help practitioners select proper metrics also taking into account the available tool infrastructure. That is, practitioners shall be enabled to select metrics and tune the selected metrics, such that they can implement these metrics with those tools that are available in the company's tool chain.

3 | RESEARCH DESIGN

Figure 2 shows the overall research methodology applied, which we describe in detail in Section 3.1. Section 3.2 presents the research objectives and research questions, before we describe the data collection procedures (Section 3.3) and the data analysis procedures (Section 3.4). Finally, we describe the procedures implemented to increase the validity (Section 3.5) of our study.

3.1 | Overall methodology

This study was conducted following a multistaged research approach in which we used a *systematic mapping study*[24] to scope the research and a *systematic literature review*[25] to perform the detailed analysis. To organize the literature studies, we followed the pragmatic guidelines defined in

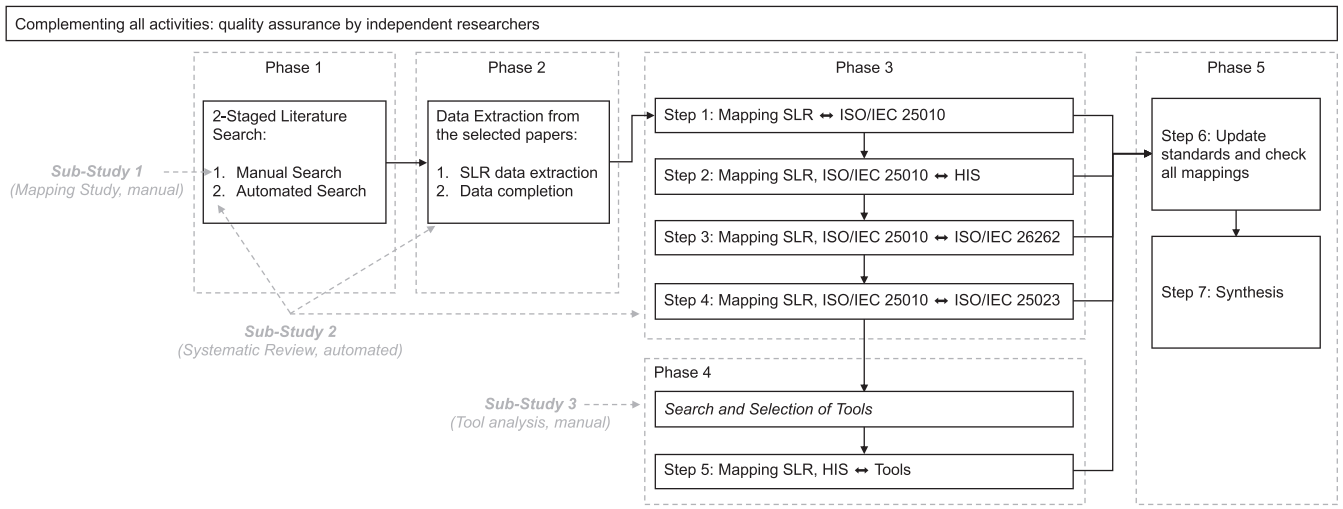


FIGURE 2 Overview of the multistaged research method (five main phases) and the three substudies implemented (a mapping study to scope the work and to develop search strings, a systematic review to collect the main data, and a collection and analysis of tools to evaluate the practical availability of the metrics)

Kuhrmann et al.[26] and adopted the "three-researcher voting model." In addition to the plain literature studies, different mappings and complementing analyses of 20 selected tools have been performed to drive the collection and structuring of the metrics obtained from the systematic review. That is, the study at hand consists of three substudies that are integrated with each other to provide a big picture. The overall research method applied, including the different analysis steps (Section 3.4), is illustrated in Figure 2 and will be explained in detail in subsequent sections.

3.2 | Research objectives and research questions

Our overall objective is to develop a catalog of metrics used in automotive software development. In this context, the quality attributes addressed by metrics, the practical use of metrics, and respective boundary values are of particular interest. To address our research objective, we defined the following research questions:RQ1: *Which metrics are reported as being used in automotive software development to evaluate quality?* Numerous metrics exist to measure software systems and help determine software quality. With this research question, we aim to collect metrics reported in literature that are (specifically) used in the field of automotive software development.

RQ2: *Which quality attributes are addressed by the metrics?* As metrics are used to support determining software quality, we are interested in the relation of the different metrics to the quality attributes of a software system. That is, we aim at answering the question if there are metrics specifically addressing certain quality attributes. For this, we utilize the three norms and standards HIS, ISO/IEC 25010:2011, and ISO/IEC 26262:2018, and we provide a mapping between metrics and these standards.

RQ3: *Which boundary values exist for the different metrics?* Basically, a metric is a function mapping a software characteristic to a number, which is used to evaluate the characteristic of interest.[6,7] However, quite often, it remains unclear when a measurement outcome of a specific metric can be considered good or bad. In this research question, we analyze the metrics obtained from the systematic review for boundary values and study the contexts in which such boundary values are defined.

RQ4: *How are metrics implemented and supported in software development practice?* Finally, we are interested in the applicability of metrics, notably the tools and the language families for which the metrics provide support. The goal of studying this research question is to investigate how far metrics described in scientific literature are practically implemented in tools at the market and how the availability of tool-supported metrics impacts the selection of proper metric sets in projects.

3.3 | Data collection procedures

To collect and select the papers of interest, we adopted the "three-researcher voting model."[26] Figure 3 refines the first phase of our study (Phase 1; two-staged literature search, Figure 2) and illustrates the overall data collection approach, which consists of a manual keyword-based web search for reference papers, which are listed in Table B1, and an automated search in the Web of Science (WoS; core collection), which yielded another 26 papers listed in Table B2. The subsequent sections provide details on the search procedure.

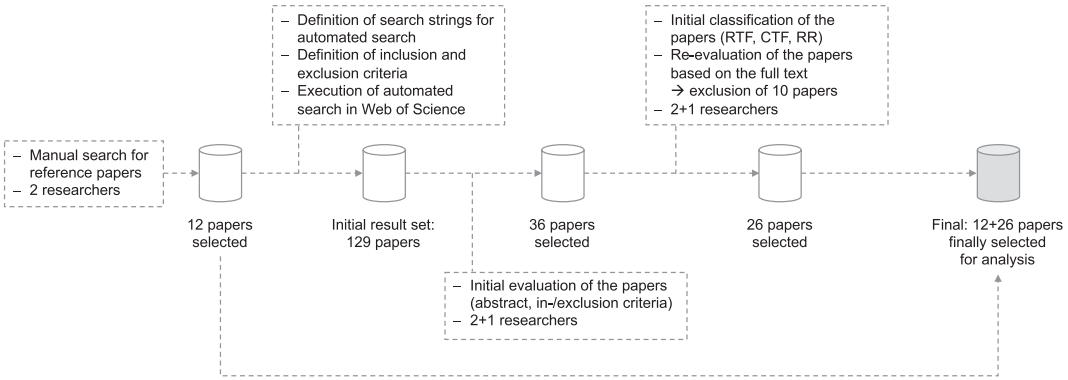


FIGURE 3 Data collection method implemented in the literature review

3.3.1 | Query string construction and automated search

As the purpose of the study was to collect and systematize metrics for automotive software development, we opted for an opportunistic query construction approach. Initially, we defined the search string *("safety" OR "automotive") AND ("software metrics" OR "software metric")*, which was the basis for the query construction. The goal of this search string was to get a set of publications dealing with metrics in automotive software development or, at least, general metrics applied to safety-critical system development. To test and refine the search,[26] we used the initial search string and several variants with Google Scholar (executed on June 29, 2018) to generate "test" result sets, which we analyzed for suitability, that is, if they contain previously defined reference papers (Table B1). To analyze the suitability of a search string, we used the keywords of the returned papers to create word clouds,[26] which we visually inspected for a sufficient coverage of the topics of interest and an acceptable overhead (papers outside the area of interest). The word clouds provided the frequency of the keywords, and we collected those keywords that had a minimum frequency of three and were in scope of the study. The resulting list of keywords was consolidated, and the consolidated keyword list was used to create keyword groups, which were used to derive the final search strings shown in Table 1.

Eventually, we concluded the search strings listed in Table 1, which were constructed to generate overlapping result sets in order to minimize losses as described by Kuhrmann et al.[26] The expected multiple occurrences of papers have been resolved in the dataset cleaning and the final paper selection procedures (Section 3.3.2). To avoid the need of implementing extra activities to resolve "cross-database-cross-search" issues, that is, multiple occurrences of papers in the dataset due to the overlapping search strings applied to multiple digital libraries, we opted for a meta-search engine to execute the search. The search was designed as a topic-based search, which already includes the fields title, abstract, author keywords, and keywords plus, using the WoS (core collection). The search using the WoS was executed on July 27, 2018 and yielded 143 papers in total. As Table 1 shows, 14 duplicates (<10% overhead) have been removed from the result set such that 129 papers remained for evaluation. For each hit in the WoS database, the full reference (including author list, title, abstract, and so forth, see Table A1) was exported into a spreadsheet file.

TABLE 1 Final search strings for the automated search in the Web of Science and the number of papers returned by the respective strings including duplicates among all the different search strings in the whole result set

ID	Search string	Papers found	Duplicates
1	TOPIC: automotive AND software AND metric$	29	1
2	TOPIC: safety AND automotive AND software AND metric$	5	5
3	TOPIC: introduction[a] AND software AND metric$	102	1
4	TOPIC: framework AND automotive AND software AND metric$	5	5
5	TOPIC: (ISO 26262 OR ISO26262) AND software AND metric$	2	2
	Total (including duplicates):	143	14
	Total (cleaned):	**129**	

[a]The term "introduction" was included to get basic literature on (general) software metrics into the result set. The purpose of looking for introductory work was to obtain definitions of metrics in case special literature only names a metric but provides insufficient details.

TABLE 2 Overview of the inclusion criteria (IC) and the exclusion criteria (EC)

ID	Criterion
IC1	Paper is on automotive software and metrics
IC2	Paper is on metrics in dependable systems domain (as defined by Avizienis et al.,[27] not necessarily automotive)
IC3	Paper is a secondary study on metrics in dependable or automotive software development
IC4	Paper contains quality models or a general measurement approach but in relation to dependable systems
EC1	The paper is not on the domain of automotive software[a]
EC2	The paper is a workshop summary, a guest editor introduction etc., i.e., the paper is not an original research article
EC3	The topic of interest is only mentioned in the introduction or related work, but is not a key contribution of the paper
EC4	The paper is not available for download

[a]Note that the exclusion of nonautomotive software papers was part of the cleaning procedure. However, if papers were identified that describe metrics outside the field of interest, usually, such papers have been considered for complementing data extraction and completion of data points.

3.3.2 | Inclusion and exclusion criteria, search execution, data collection and extraction, and evaluation

To select the papers for the study, we defined the inclusion and exclusion criteria shown in Table 2. The evaluation of the result set using the inclusion and exclusion criteria was independently performed by two researchers. A third researcher evaluated the two votes and created a new dataset from the results of the individual votes.[26] Using the inclusion and exclusion criteria, all papers were evaluated for inclusion or exclusion in the study, and if a paper was accepted, the required data for the analyses were extracted. Finally, an integrated spreadsheet file was created that included all individual reviewer votes and extracted (meta) data. The integrated spreadsheet consisting of 36 candidate papers was re-evaluated by the whole team for finally selecting the papers to be included in the study. Eventually, 26 papers from the automated search have been selected for inclusion (Table B2). Together with the 12 handpicked reference papers (Table B1), the final dataset for the study consisted of 38 papers.

Besides the metadata (Table A1), for each paper, the data extraction (Figure 2; Phase 2) was performed using the structure shown in Table A4. In the course of executing the different data collection, extraction, and analysis steps, the data structure evolved, for example, by adding extra attributes and scores. Specifically, in the first iteration, we extracted all metrics mentioned in the respective papers (Table D1). In further iterations, we completed and extended the data, before, in the final iteration, we cleaned up and checked the extracted data to ensure that a metric is present in the dataset only once and all information regarding a specific metric was consolidated. Furthermore, during the data extraction, we learned that detailed information, for example, about the formulae used to compute the metrics, was available for few metrics only. Specifically, the papers obtained in the search yielded 39 formulae, 51 metric descriptions, 12 boundary values, and 19 mappings between metrics and quality attributes. Therefore, we used the European Space Agency's (ESA) ECSS standard[17] and further *gray literature*[28] (focused search for content in other literature that was not included in the 38 papers) obtained in a *snowballing procedure* as described by Badampudi et al.[29] to complement our data (Table A4) with detailed information. This extra literature increased the quality of the information, such that—after integrating it—in total, 112 metrics and their descriptions were available for analysis, that is, for the extraction of formulae, the identification of boundary values, and for performing the various mappings.

3.4 | Analysis procedures

To analyze the papers, we defined a basic data structure that includes the attributes of interest (Table A4). On the basis of the extracted data, we performed the analyses using the analysis model shown in Figure 4. The analyses included a number of mappings and comparisons of the data extracted from the systematic review with different standards relevant to the field of interest (cf. Figure 2; Phases 3 and 4). Specifically, we investigated the coverage and application of metrics found in the systematic review in the context of the standards introduced in Section 2.1.

All analysis steps were initially conducted by the academic researchers in the team, who presented the (tentative) findings in workshops and weekly phone calls to the practitioners in the team. In these workshops and calls, results have been discussed and next steps for the analysis were defined. Finally, as illustrated in Figure 2 (Phase 5), all results were checked again. These final checks also included an update of the standards to their latest versions[‡]and a re-evaluation of the study's findings in the context of the new standards before executing the synthesis.

[‡]During the data analysis steps, the standard ISO/IEC 26262 was updated, and the industry partners made the new version available to the team. After a discussion on how to treat the new version, the actual analysis was paused, the new standard was analyzed, and, eventually, the analysis steps executed so far were repeated with the new standard version ISO/IEC 26262:2018 as a new baseline.

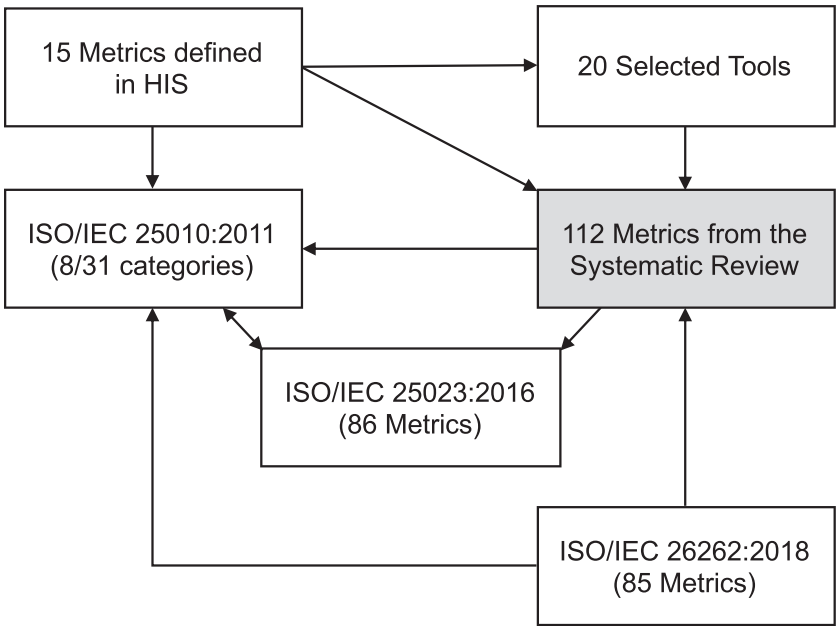


FIGURE 4 Analysis model applied to the dataset

The starting point in our analysis model from Figure 4 are the 112 metrics identified in the dataset (in the following, we just speak of *SLR-metrics*, see Table D1). In the next step, the 15 metrics defined by HIS metric catalog were mapped to the 112 SLR-metrics (the metrics either matched directly or the description of the metrics matched to a large extent), which created a unidirectional link between the HIS metrics and the 112 SLR-metrics. After linking the HIS metrics and the SLR-metrics, every SLR-metric and every HIS-metric was mapped to one or more of the 31 subcategories in ISO/IEC 25010:2011 (Figure 1). In the next step, the ISO/IEC 26262:2018[4] (Part 6) was analyzed for metrics, which were mapped to the SLR-metrics as well. In case a mapping was not straightforward, the problematic metric was discussed in the team workshops. To round out the picture, in the last step, the ISO/IEC 25023:2016[9] was mapped to the SLR-metrics. Because the ISO/IEC 25023:2016 provides a mapping to ISO/IEC 25010:2011, the finally developed mappings allow for putting metrics into context, such that a metric from the HIS metric catalog can be positioned in the ISO/IEC 25010:2011 standard and so forth.

Finally, to study the practical relevance of the metrics, in the team workshops, it was decided to also include tools used for collecting and analyzing metric data (Section 3.4.2). The procedure to link metrics to tools was implemented the same way the metrics' mapping was performed. That is, the HIS-metrics and the SLR-metrics were mapped to tools to study which metrics are supported by the tools.

In the following, we describe the two core analysis steps: (i) the mapping of the metrics with the quality attributes as defined in the ISO/IEC 25010:2011 and (ii) the mapping of the metrics with selected tools.

3.4.1 | Analyzing and mapping metrics with quality attributes

Because the HIS metric catalog is still frequently used in the article's industrial context, we started with analyzing how many of the HIS-metrics (Section 2.1) were found in the systematic review and which parts of the HIS metric catalog are covered. To link the HIS-metrics and the SLR-metrics, we developed a mapping from the HIS-metrics to the ISO/IEC 25010:2011 quality attributes, before we performed a mapping from the SLR-metrics obtained from the systematic review to the ISO/IEC 25010:2011 quality attributes and the associated metrics defined in the ISO/-IEC 25023:2016.[§]As for some metrics an appropriate mapping was provided in the respective papers (Appendix C1), we collected this information and completed the mapping ourselves where necessary. Finally, we analyzed all metrics for their level of support through tools (see Section 3.4.2).

3.4.2 | Analyzing and mapping metrics with tools

To analyze the tool support for metrics, we selected 20 tools that we analyzed for the metrics they support and for the coverage of the SLR-metrics. Specifically, we analyzed if a specific metric is fully supported by tools, that is, if data collection and evaluation is fully automated, if

[§]Please note that the ISO/IEC 25023:2016[9] covers more metrics than defined in ISO/IEC 25010:2011. Furthermore, as we were interested in analyzing the coverage of metrics in detail, our analyses were performed using the 31 detail-level criteria form ISO/IEC 25010:2011, see Figure 1. Also, the ISO/IEC 25023:2016 also provides basic formulae and thresholds. These were, however, *not* used in the article at hand as we were explicitly interested in the practically defined/used thresholds reported in literature.

TABLE 3 Search strings applied to the web search for tools that support software measurement and overview of the resulting tools after applying the exclusion criteria from Table 4

ID	Search string	Initially selected tools
1	"simulink" and "metric"	Simulink Check
2	("tools" OR "tools * code") AND ("analyzer" OR "analysis")	Gamma, Sonargraph, NDepend, Frama-C, PMD, SourceMeter, Blu Age Analyzer, Parasoft C/C++test, Resource Standard Metrics (RSM), Designite
3	"Eclipse * code" AND ("analyzer" OR "analysis")	Eclipse Metrics
4	"software code" AND ("analytics" OR "analysis")	–
5	"code metrics" AND ("analytics" OR "analysis")	–
6	"software code" AND ("metrics tool" OR "metric tool")	–
	Total:	**12**

TABLE 4 Overview of the exclusion criteria for tools (EC_T) that were used to reject tools returned by the web search

ID	Criterion
EC_T1	The tool is in the status "support canceled"
EC_T2	The tool's documentation does not contain any information about the implemented/supported metrics
EC_T3	There is no publicly accessible documentation for the tool available
EC_T4	There is no test/evaluation version of the tool available

there is a partial support, that is, at least some steps in the data collection and evaluation are automated, or if the metric has no tool support, that is, all steps in the data collection and evaluation have to be done manually. To find relevant tools, we conducted a web-based manual search using the search criteria summarized in Table 3. The purpose of this search conducted by the researchers in the team was to create an initial set of tools to be discussed in the whole team. After applying the exclusion criteria shown in Table 4, 12 tools were initially selected (Table 3). Note that the resulting tools are counted including their language variants, that is, SourceMeter is available for Java, C++, and so forth, but generates only one hit in Table 3. However, during the analysis, the language variants have been analyzed as well. The tools were used as additional input for assessing the practical relevance. It has been assumed that metrics found in the SLR and supported by many tools are of more practical relevance.

To confirm the practical relevance of the selected tools, the initial set of 12 tools was presented to the practitioners in the team who were not involved in the search for tools. On the basis of the discussion, the tool set was extended to include the tools *Development Assistant for C* (DAC), *Klocwork*, and *Polyspace Code Prover* (QA-C), which are frequently used in the company's tool chain but were not present in the initial list of tools in Table 3. After extending the list of tools, the whole team discussed the extended list again and agreed on the appropriateness for the analysis. The finally analyzed 20 tools (including variants) are listed in Table 6, which also shows the different metrics found in the systematic review and how these metrics are supported by the selected tools. However, it has to be noted that this search is scoped to general tools that are available for analysis and those tools specific to the industrial context of the practitioners in our team. This introduces a threat to validity, which is discussed in Section 5.3.

3.5 | Validity procedures

To constructively improve the validity of our findings, we implemented several procedures. First, we rely on available standard procedures to define the study and collect and analyze the data. We framed the study by analyzing the input material (HIS, Section 2.1) and discussing the findings in workshops (academic and industrial partners). Finally, we conducted a mapping study according to Petersen et al.[24] using a limited set of the 12 handpicked papers that serve as reference papers (cf. Figures 2 and 3). After analyzing the reference papers, a full systematic review according to Kitchenham et al.[25] was conducted following the work mode as described in Kuhrmann et al.[26]

To ensure the rigor of the individual steps and to improve the validity of the results, we established a team-based working style. That is, the team of researchers was split such that at least two researchers performed an actual activity, for example, collecting the data, analyzing the data, and so forth. From the remaining researchers, at least one not involved in the respective activity conducted a quality assurance. In case of stalemates during evaluation processes, a researcher not involved in the decision-making was called in to evaluate the critical object and to decide. In general, evaluations were independently performed by at least two researchers, and a third researcher integrated the results. As illustrated in Figure 2, all activities were continuously quality assured. For this, we defined a work mode in which the academic researchers involved in the study were mainly concerned with analysis and synthesis tasks whereas the practitioners participating in the study performed continuous quality assurance. In workshops and in weekly phone calls, tentative results, problems, and issues were discussed, and further study activities were defined.

4 | STUDY RESULTS

This section presents the findings of our study. We start with a result set overview in Section 4.1 before we present the findings structured according to our research questions as presented in Section 3.2.

4.1 | Overview of the result set

As described in Section 3.3.2, our study finally included 38 papers in total. Figure 5 shows the publication frequency of the papers analyzed. Detailed information about the papers included in the study can be taken from Appendix B1.

To better characterize the analyzed papers, we classified the papers according to the *research type facet* (RTF; Wieringa et al.[30]) and the *contribution type facet* (CTF; Shaw[31]), which is illustrated in Figure 6. The categorization in Figure 6 shows our result set providing a considerable share of solution proposals and lessons learned. Yet, we also find models and theories proposed in the result set. In summary, approximately two thirds of the papers propose solutions (Figure 6, dimension RTF) and approximately one third of the papers reports on lessons learned (Figure 6, dimension CTF). Also, the result set contains only one paper describing a tool-based solution.

An evaluation of the papers according to the *rigor-relevance model* by Ivarsson and Gorschek[32] is illustrated in Figure 7. The evaluation shows that 13 out of the 38 selected papers (upper right quadrant of Figure 7) are evaluated to be of high to very high relevance (score ≥3) and of high rigor (score ≥2.5). Only seven papers received an evaluation of rigor ≤1.0, and 12 papers in total received an evaluation for the relevance ≤1 of which six papers are in the lower left quadrant. Hence, we consider the overall result set of sufficient practical relevance, and we consider the included papers as having undergone a research procedure of sufficient rigor.

4.2 | RQ1: Which metrics are reported as being used in automotive software development to evaluate quality?

The first research question is concerned with identifying the metrics used in automotive software development as reported in literature. In total, the 38 analyzed articles provided 112 metrics that address various artifacts and that can be applied to various programming languages. The complete catalog of metrics identified in the systematic review can be taken from Table D1. To categorize the metrics, in a team workshop, we reviewed all metrics and agreed on the following three categories (adapted from Kan[8]):

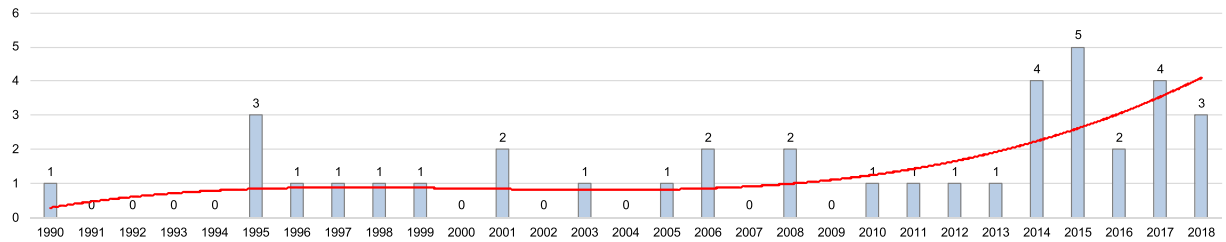


FIGURE 5 Publication frequency of the papers in the result set

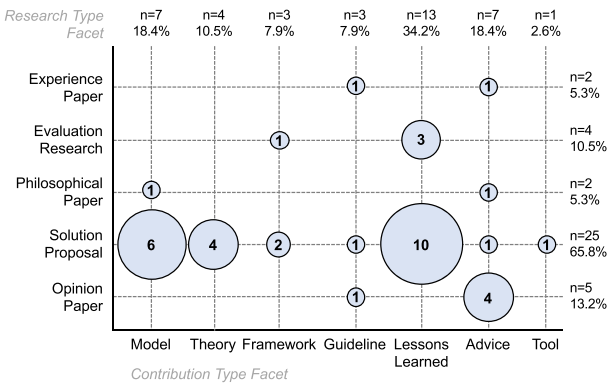
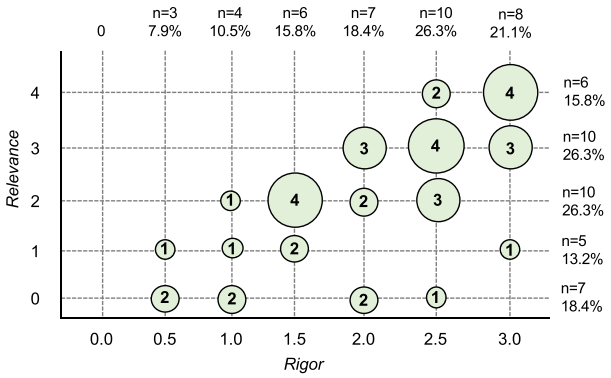


FIGURE 6 Categorization of the papers according the research type facet and the contribution type facet

FIGURE 7 Categorization of the papers according the rigor-relevance model



1. *Process metrics.* This category includes metrics to quantify properties of the software development process.
2. *Product metrics.* This category includes metrics to quantify properties of the product as such, for example, represented by models and architecture documentation.
3. *Code metrics.* This category includes metrics to quantify properties of source code.

Figure 8 shows the assignments of the 112 SLR-metrics to the three metric categories. In total, we made 124 assignments for the 112 metrics. The figure shows that the majority of the found metrics is concerned with source code, models, and architecture. Metrics applied to the measurement of binaries are not present in the result set. Also, most metrics are categorized as code metrics and product metrics, which is in line with previous studies.[13,15]

Because we were interested in the information provided alongside the metrics' naming, we analyzed the papers for the availability of formulae used to compute the metrics. In total, for 54 out of the 112 SLR-metrics, we could find formulae or algorithms, for instance, simple formulae, for example, *Lines of Code* (LoC; Metric ID 2 in Table D1), complex formulae, such as the *Component Input Complexity*[33] (Metric ID 27 in Table D1), or algorithms, for example, the *Fault Coverage*[34] (Metric ID 60 in Table D1). For another eight metrics, we utilized "gray literature" to add formulae to the metrics (e.g., the *Classified Attributes Inheritance* (CAIW), named in Mumtaz et al.[35] and a formula found in Alshammari et al.[36]; Metric ID 19 in Table D1). Finally, for 50 out of 112 metrics, we could not find proper information regarding the metrics' structure or their computation.

Finding 1: Most of the metrics found in the systematic review are concerned with source code (e.g., *McCabe*, *Lines of Code*, *Henry and Kafura*, *Halstead*). Product metrics as the second-ranked category are mainly concerned with models and architecture descriptions (e.g., *Ease of function learning*, *Function points*, *Traced Components per Requirement*). Metrics applied to binaries are not contained in the result set. Another finding is that only a few metrics have been found explicitly addressing automotive software development. It was found in the analysis that there are many cross-sectional relationships to the dependable systems domain. In total, only 54 formulae could be extracted from the papers included in the systematic review, and another eight formulae have been added through studying "grey literature". That is, 50 metrics were mentioned in the text only without further details (e.g., *Number of Statements*).

4.3 | RQ2: Which quality attributes are addressed by the metrics?

A key question in our research is concerned with the quality attributes addressed by specific metrics. That is, we analyzed which metric or set of metrics shall/can be applied to assess a specific quality attribute of a software system.

According to our analysis model from Figure 4, we initially mapped the metrics to the different standards as described in Section 3.4. Figure 9 provides an integrated perspective on the mappings for which the detailed mapping tables can be taken from Appendix C1. We mapped the metrics to the ISO/IEC 25010:2011 to check the general coverage of the quality attributes addressed by the SLR-metrics found in the systematic review. Specifically, we studied the mappings provided by the two ISO standards 25010:2011 and 25023:2016, which is illustrated in Appendix C1 (Figure C1). The ISO/IEC 25023:2016 provides 86 metrics for which a mapping to the 31 subcategories of the ISO/IEC 25010:2011 product quality model (Figure 1) is provided. We used this standard mapping as a baseline for mapping the SLR-metrics to the different standards. We mapped the SLR-metrics to the HIS metric catalog and to the ISO/IEC 25010:2011 quality attributes to assess the coverage. The mapping to the HIS-metrics is provided in Table C1, and the mapping to the ISO/IEC 25010:2011 quality criteria is provided in Table C2.

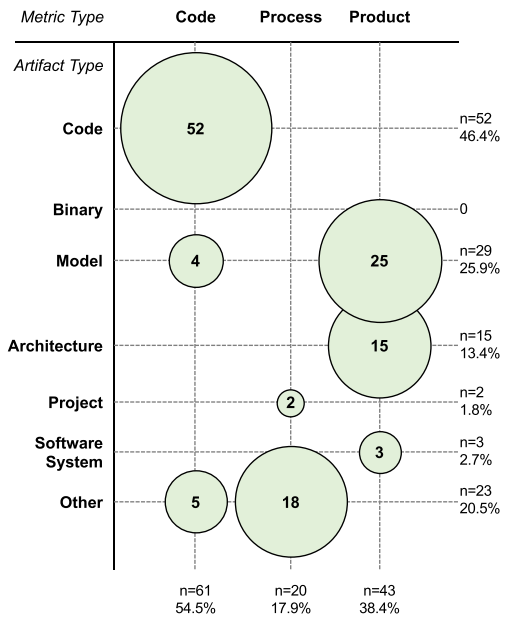


FIGURE 8 Categorization of the metrics according to the artifacts addressed. Note that a metric can address more than one artifact, i.e., all numbers refer to the 112 SLR-metrics

The mapping to the HIS metric catalog shows that all but one metric defined in the HIS metric catalog could be found in our result set. Only for the HIS-metric *Number of Recursions* could no proper mapping be found. The mapping to the ISO/IEC 25010:2011 in Table C2 shows that we found a good coverage but that there is a specific focus on the quality attributes of the top-level category *Maintainability* (Figure 1), which contains the quality attributes *Modularity*, *Reusability*, *Analyzability*, *Modifiability*, and *Testability*. The third standard of interest is the ISO/IEC 26262:2018, which is specifically designed to support the development of safe road vehicles. To provide a big picture in which we combine the mappings of the HIS metrics and the ISO/IEC 25010:2011 quality attributes with the ISO/IEC 26262:2018, Figure 9 provides an integrated perspective. This mapping shows that the majority of the metrics addresses the category *Maintainability*. Also note that Figure 9 is a reduced presentation based on available mappings of the metrics found in the systematic review and their assignment to quality attributes to allow for providing an integrated perspective. The full mapping of the ISO/IEC 26262:2018 to the ISO/IEC 25010:2011 quality attributes can be taken from Appendix C1 (Figure C2). Further detailed assignments from the considered standards and the metrics obtained from the systematic review can be found in Tables C3 and C4.

Finding 2: Most of the SLR-metrics address the ISO 25010 top-level category *Maintainability*, which contains *Modularity*, *Reusability*, *Analyzability*, *Modifiability*, and *Testability*. For some quality attributes, our study did not provide metrics to cover all subcategories of the ISO 25010. The different mappings in Appendix C also reveal that certain system characteristics are emphasized while others have no proper metrics defined.

4.4 | RQ3: Which boundary values exist for the different metrics?

A software metric is a function whose inputs are software data and whose output is a numerical value that can be interpreted as the degree to which software possesses a given attribute that affects its quality.[37] However, quite often, it remains unclear when a measurement outcome of a specific metric can be considered good or bad. For this, *boundary values* are introduced that support the interpretation of a metric.

Our third research question is concerned with such boundary values and if/how they are defined in the found metrics from the systematic review and in the standards included in the study. Table 5 shows that 17 out of the 112 SLR-metrics define boundary values. Of these 17 metrics, 11 boundary values were found in the result set of the systematic review, and six have been added by studying the external ECSS-Q-HB-80-04A standard.[17] The external ECSS standard was included in the study, because this standard also addresses dependable systems and provides a set of metrics including boundary values and application scenarios comparable to the ASIL levels defined in the ISO/IEC 26262:2018.[4] Please also note that the extraction of boundary values did *not* include the boundary values as defined in the ISO/IEC 25023:2016[9] as these focus on product quality in general. Yet, as the primary goal was to analyze the state of practice as reported by the papers on automotive software development or safety-critical systems in general, the ISO/IEC 25023:2016 thresholds have not been included (see also Section 3.4.1).

TABLE 5 Thresholds of metrics obtained from the systematic review and extended by the ECSS standard; note that the minimum and maximum values found only indicate boundary values, which are, however, specific to the actual context and, thus, do not allow for absolute statements

MetricID	Paper ID	Metric	Min	Max	From SLR	From ECSS
1	38-42	McCabe cyclomatic complexity number	1	X		✓
2	39-43	Lines of Code (LOC)	0	75		✓
10	39	Belady's bandwidth (nesting levels)	1	T	✓	
11	38	depth of nesting-DON	1	T	✓	
12	35	Lack of Cohesion in Methods (LCOM)	40	108		✓
26	33	complexity of a single component	>0		✓	
35	35,42	Coupling between objects (CBO)	0	25		✓
51	35	Depth of Inheritance Tree (DIT)	2	6		✓
53	44	Ease of function learning	0	D	✓	
55	45	Error-detection	0	1	✓	
69	44	Interface appearance customizability	0	1	✓	
85	35	Number of Children (NOC)	0	6		✓
98	35	Response for a Class (RFC)	34	74	✓	
99	44	Self-explanatory error messages	0	1	✓	
109	44	Understandable input and output	0	1	✓	
110	44	Usability compliance	0	1	✓	
111	35	Weight Methods for Class (WMC)	0	26	✓	

Note: X = maximum number of paths (natural number); T = number of maximum nesting; D = mean time taken to learn use a function correctly (can not represent the maximum); MetricID 26: >0 copy from text; explicit specification.

Finding 3: Only 17 out of 112 analyzed SLR-metrics define boundary values. From these 17 boundary values, only 11 could be found in the result set of the systematic review and another six were added by the external ECSS standard. If available, boundary values are also defined for different application scenarios, i.e., boundary values are specific to particular contexts and project setups or even for different ASIL levels. This fact makes it dicult to define uniform and comparable boundary values for a domain.

4.5 | RQ4: How are metrics implemented and supported in software development practice?

Finally, we are interested in the applicability of the metrics in the software life cycle. In this regard and in the context of the study at hand, we define *applicability* as the availability of a software tool that helps collecting data and computing a metric. As most of the metrics found in the systematic review are categorized as code metrics (Figure 8), we analyzed the result set for these code metrics. Specifically, we analyzed 20 free and commercial tools used in the automotive sector investigating which of the metrics obtained from the systematic review are supported by the tools (cf. Section 3.4.2). The analysis was conducted in two steps. The first step was concerned with extracting the programming languages for which the metrics provide support, and the second step was concerned with studying the tool support for the metrics.

Figure 10 summarizes the findings of the first analysis step. Assignments were made (i) by categorizing a metric whether it is a code-related metric or not and (ii) by categorizing a metric according to (programming) language families. Figure 10 shows that 128 assignments[¶] in total were made in the category for code-related metrics of which 21 assignments are not bound to a specific programming or modeling language. As the figure shows, the most assignments were made for object-oriented programming languages (52 assignments), followed by the procedural programming languages (28 assignments), which are still very popular in automotive software development. Our result set also includes metrics that can be applied to modeling languages such as VHDL and Simulink. As the number of assignments shows, several metrics can be applied to

[¶]Note that a metric could have multiple assignments. The 128 assignments are based on the initial classification from Figure 8. Yet, in the detailed analysis, further assignments have been made on the basis of available information about documented use in supported programming languages. That is, if the documentation of a metric states that, e.g., this metric is applicable to Java and C ++, this generates two assignments in Figure 10.

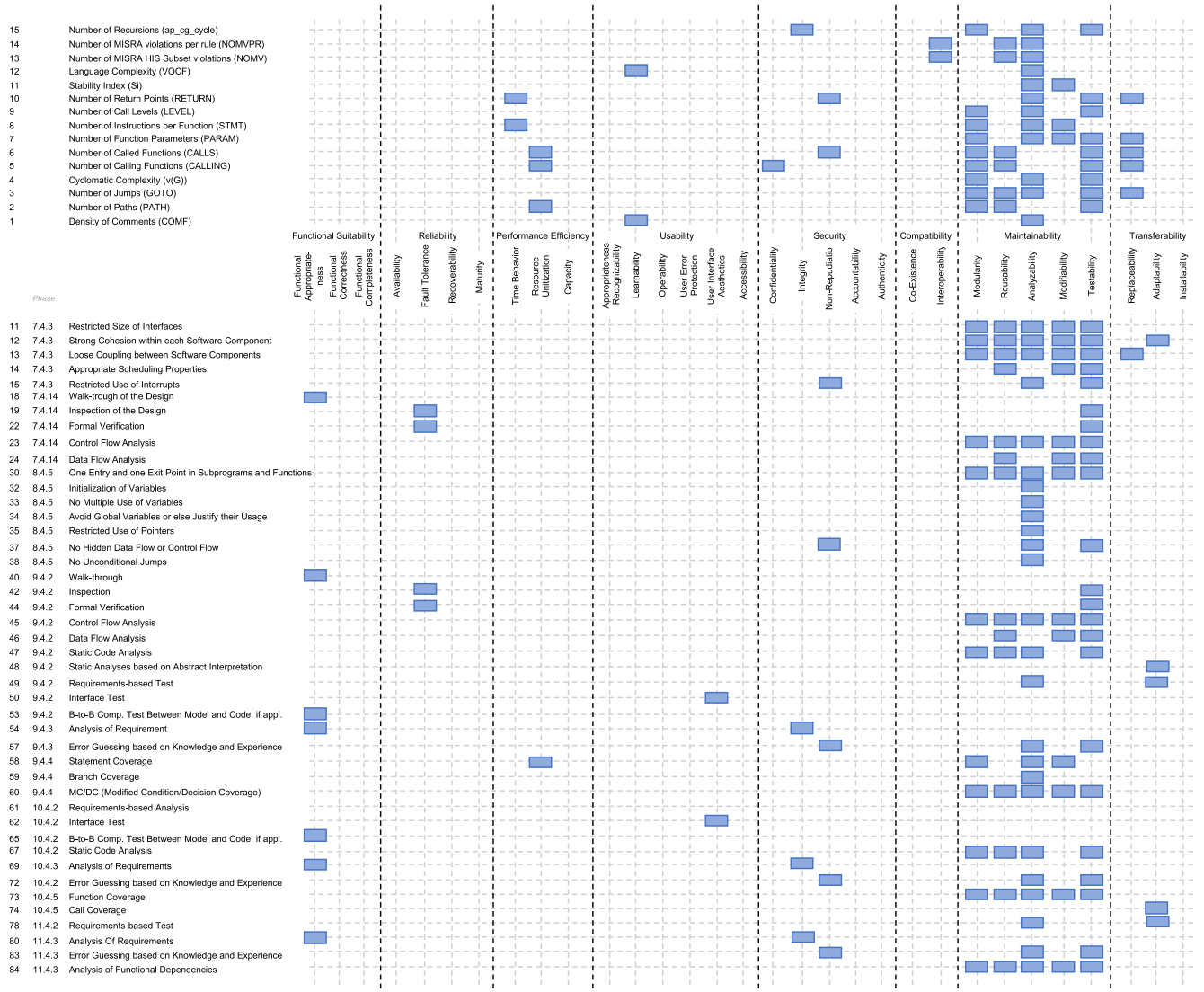


FIGURE 9 Integrated perspective on the metrics defined in the HIS (top) and the ISO/IEC 26262:2018 phases (bottom; only those phases that include the metrics of interest) and their coverage of the quality attributes as defined by the ISO/IEC 25010:2011. The mapping is shown through the blue boxes, which illustrate that a metric is assigned to a quality attribute

multiple categories. For instance, the metrics *Directed Dependency between Software Components* (52) or *Number of Statements* (91) can be applied to numerous programming languages, which could also be a reason for lacking boundary values for these metrics as stated in Section 4.4.

The second step of the analysis was focused on the selected tools as such. Table 6 lists the 20 tools we analyzed and presents the number of metrics these tools already include (built-in metrics). In the context of our study, we studied which of the metrics from the systematic review are supported by the tools listed in Table 6. For this, Table 6 provides two assignments: the first assignment is between the tools and the SLR-metrics. The second assignment is between the HIS-metrics and the tools studied.

Figure 11 provides a visual representation of the tools' coverage of the SLR-metrics. The figure shows that the tools provide a variety of metrics and that over 40% of the SLR-metrics are supported by these tools. In total, we found 48 SLR-metrics supported by the selected tools,

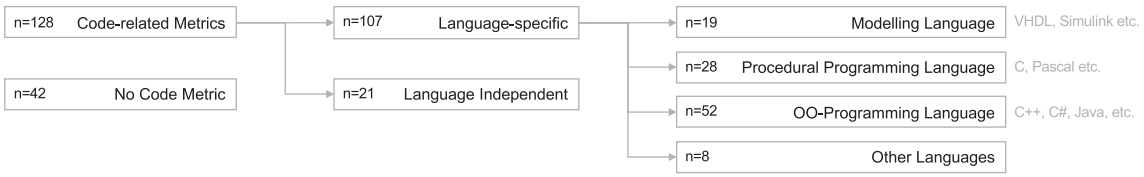


FIGURE 10 Language families (modeling, programming, etc.) supported by the metrics obtained from the systematic review

TABLE 6 Tools analyzed and assignment of tool-supported SLR-metrics and HIS-metrics (note that the metric IDs refer to the unique metric ID for the SLR-metrics (Table D1) and, for the HIS-metrics, to the HIS-index from Table C1)

			Metrics		
ID	Tool	Ref	Built-in	From SLR (Table D1)	HIS-metrics supported (Table C1)
1	Eclipse Metrics	[46]	14	1, 2, 10, 12, 18, 91, 111	4,9
2	Simulink Check	[47]	26	1,2,3,48,14,95,102	3, 4, 6, 10
3	Blu Age Analyzer	[48]	5	2, 1, 74	4, 7
4	Gamma	[49]	22	1,35,51,24,98,2,4,12,111,52,17	1, 4, 5, 8, 11, 12
5	Parasoft C/C++test Professional	[50]	49	35,1,4,12,2,98,52	1, 4, 8, 11, 12
6	Resource Standard Metrics (RSM)	[51]	5	2	
7	Sonargraph	[52]	109	1,3,2,91,30,31,33,34,13,92,37,46,40,39	6, 7
8	DesigniteJava	[53]	13	2, 1, 111, 85, 51, 12	4
9	Designite	[54]	19	2, 1, 111, 85, 51, 12	4
10	SourceMeter 8.2 for Java	[55]	98	4, 1, 111, 35, 98, 51, 85, 2, 91, 11, 52	1, 4, 5, 6, 8 , 9 , 11, 12
11	SourceMeter 8.2 for C/C++	[56]	88	4, 1, 111, 35, 98, 51, 85, 2, 91, 52, 70, 104	1, 4, 5, 6, 8 , 9 , 11, 12
12	SourceMeter 8.2 for Python	[57]	41	111, 35, 98, 51, 85, 2, 91, 11, 52	5, 6, 9
13	SourceMeter 8.2 for RPG	[58]	34	35, 2, 91, 11, 52	5, 6, 9
14	nDepend	[59]	18	2, 4, 60, 80, 82, 36, 37	1, 8, 11, 12
15	PMD	[60]	10	1, 2, 111, 17	2,4
16	Frama-C (Metrics plug-in)	[61]	12	1, 2, 4	1, 3, 8, 10, 11, 12
17	Development Assistant for C (DAC)	[62]	11	2, 91, 64, 1, 4, 81	1, 4, 8, 11, 12
18	Klocwork	[63]	118	2, 20, 4, 1, 91, 56, 49, 38, 57, 11, 50	1, 2, 4, 6, 7, 8, 9, 11, 12
19	Polyspace Code Prover	[64]	35	1,2, 35, 52, 74, 109, 14	1, 2, 3, 4, 5, 7, 10, 15
20	QA-C	[65]	69	1,98,4,14,3,2,64,10,112, 35, 109, 74	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 15

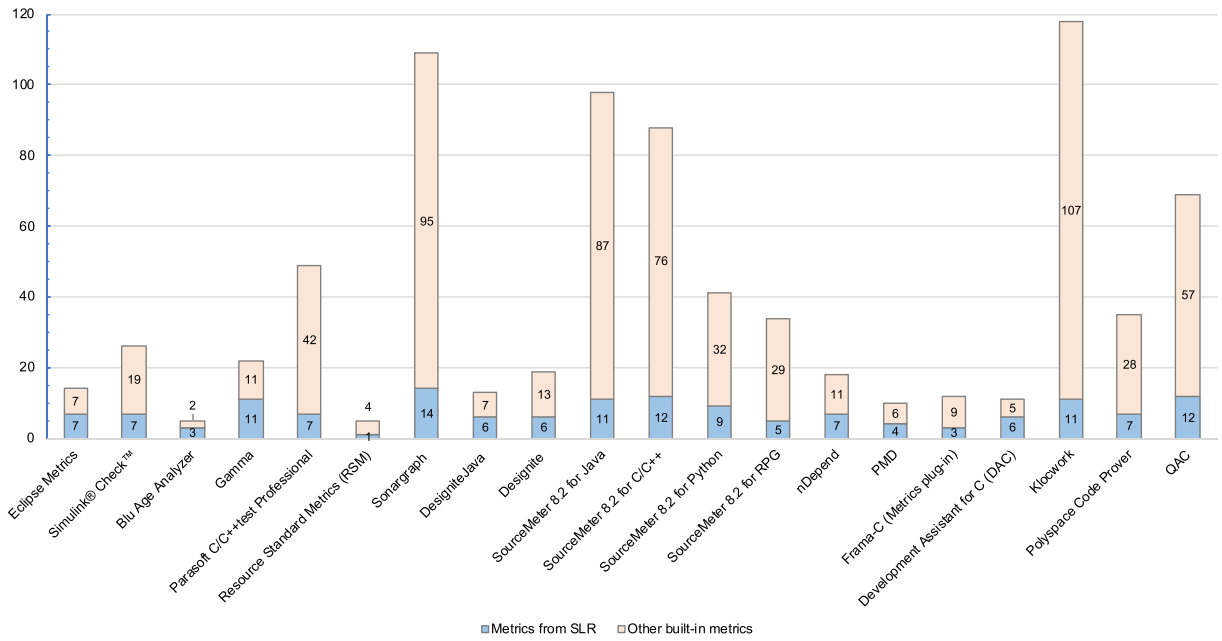


FIGURE 11 Number of the supported metrics from the systematic review in relation to the total number of metrics provided by the analyzed tools

yet not a single tool supports all of these metrics. To study which metrics are supported the most, Figure 12 provides an overview of the 48 out of 112 SLR-metrics that are supported by at least one of the selected tools. The detailed information on which tool provides support for which metrics can be taken from Table 6.

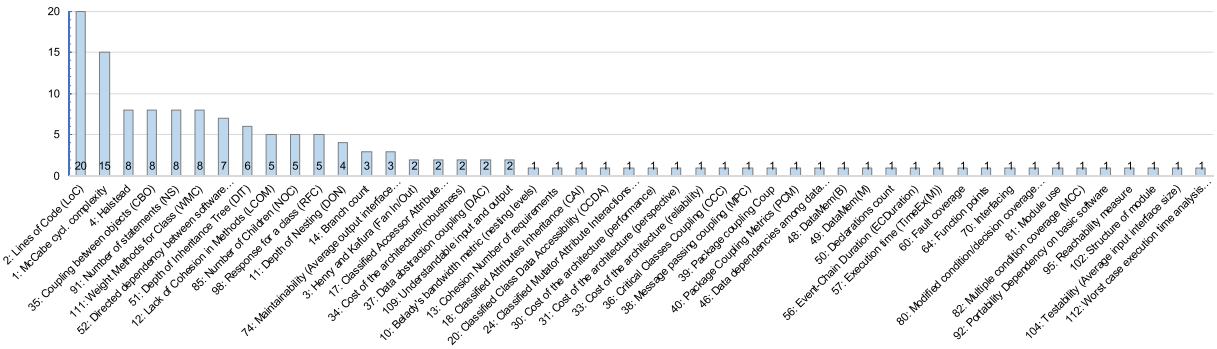


FIGURE 12 Overview of the 48 tool-supported SLR-metrics and their quantified support by the analyzed tools (how many tools support a metric)

Finding 4: Our study shows that there are metrics for which language-specific interpretations exist, i.e., that are available for different programming languages. However, only 48 out of 112 SLR-metrics are supported by the tools analyzed in this study. Even tools that provide about 100 or more built-in metrics, support a maximum 14 of the SLR-metrics. The most supported metrics are *Lines Of Code* (20 tools) and *McCabe* (15 tools). Other metrics are supported by few tools only, often by only one (specialized) tool.

5 | DISCUSSION

In this section, we discuss the findings and provide a synthesis as well as discussion of the results. Furthermore, we discuss the threats to validity of our study.

5.1 | Answering the research questions

In Section 3.2, we posed four research questions. To start the discussion of the findings of the study at hand, we wrap up the key findings and answer the research questions as follows:RQ1: The first research question is about the number of metrics reported in literature for the use in automotive software development. Our systematic review of 38 papers yielded 112 different metrics of which 61 (54.5%, e.g., *McCabe*, *LoC*, *Henry and Kafura*, and *Halstead*) were categorized as code-related metric and 43 (38.4%) were categorized as (general) product metric. Product metrics, notably, are mainly concerned with models and architecture descriptions. Also, the analyzed papers provided formulae for 54 metrics only, another eight formulae were extracted from "gray literature," but for 50 metrics, no detailed information was made available, for example, regarding the structure of the metric or a formula used to compute the metric. Among all the found metrics, however, few metrics only are explicitly mentioned in the context of automotive software development (the HIS metrics). That is, the majority of the metrics reported in practice is of generic nature, and these metrics are only interpreted for and applied to automotive software development. Table D1 provides the full catalog of metrics resulting from our systematic review.

RQ2: An assignment of the metrics obtained from the systematic review shows that most of the metrics address the category *Maintainability* of the ISO/IEC 25010:2011 standard (Figure 1). Our findings show that all subcategories as defined by the ISO/IEC 25010:2011 are covered by the SLR-metrics. Furthermore, the SLR-metrics provide an almost full coverage (only one metric was not present in the result set) of the HIS metric catalog, which is used in the German automotive software development business. Finally, our findings reveal that certain focal points are set by the SLR-metrics. For instance, whereas the ISO/IEC 25010:2011 category *Maintainability* is well-covered, other categories are not well-represented. All detailed mappings can be taken from Appendix C1.

RQ3: The third research question was concerned with the boundary values defined for the different SLR-metrics. In total, boundary values were defined for only 17 out of 112 SLR-metrics (approximately 16%). Of these 17 boundary values, 11 were defined by papers included in the systematic review, and the another six boundary values were added by including further literature in the analysis. Also, if boundary values have been found, they are mostly defined as specific to a particular project setup, that is, a set of general boundary values was not identified. Detailed information is provided in Table 5.

RQ4: Our study shows a number of metrics that exist for different modeling and programming languages. However, for only 48 out of the 112 SLR-metrics, tools that support these metrics have been found. Even those tools that support a variety of metrics support only 14 of the SLR-metrics at maximum. Whereas there are "standard" metrics like *LoC* that are supported by (almost) all tools, other metrics are, if at all, supported by only one tool. Details are presented in Section 4.5.

5.2 | Discussion

Our findings provide building blocks to discuss the use of metrics in automotive software development. Specifically, we are interested in developing proper views to help practitioners selecting subsets of metrics that provide a maximum coverage of quality attributes and that, at the same time, help in establishing a compliant measurement tool support. For this, we did not only analyze literature for the purpose of collecting metrics but also put the found metrics into context. In total, we identified 112 metrics (Table D1) in our result set. These metrics were mapped to established standards (see Appendix C1 for further details) and a number of tools used in automotive software development to collect and evaluate metrics (Section 4.5).

Our findings as summarized in Section 5.1 indicate that handling metrics with the focus on value ranges, quality attributes, and context in practice is challenging. Value ranges were defined for only 17 metrics, and no agreed (standardized) association with quality attributes was found, which is in line with findings found in related literature.[12-15,66,67] However, when performing a mapping procedure (Section 4.3), our findings show a focus on maintainability-related quality attributes. Metrics addressing quality attributes like those related to functional suitability or security (Figure 1) were not found in the studied literature. Moreover, few studies only provided information about an exact mapping between a particular metric and the quality attributes addressed. This indicates an often observed challenge in practice: misunderstandings among clients and contractors about required qualities, acceptable value ranges (including boundary values),[66,67] and what is considered good or bad, and how this perception is reflected by a metric or a metric set.[13]

As many metrics in software development are focused on code (Figure 8), using tools to evaluate software using metrics is the straightforward approach. Table 6 shows that a number of tools is available, and each tool supports numerous metrics. A finding from our study is that only 48 out of 112 SLR-metrics are supported by the selected tools. Besides the standard metrics *LoC* and *McCabe*, only few tools support the metrics found in our study, whereas many tools provide support for metrics that are not included in our metric catalog. This raises the question of whether the metrics reported in the literature we analyzed are relevant at all. However, this question has to be answered in the light of the standards applied to automotive software development (Section 2.1). The mappings performed (Section 4.3 and Appendix C1) show the found metrics properly addressing the relevant standards. This potential disparity could be caused by the way metrics and measurements are implemented in practice using a tool or a tool set as facilitator. That is, we argue that when a tool is deployed to a company, those metrics from the tool's built-in catalog that can be implemented with the least possible effort will be chosen to implement the measurement processes. Other metrics or a measurement program that puts quality attributes (from which the metrics of interest have to be derived) in the spotlight challenge the companies, which might be a reason for the difficulties companies have to adapt more comprehensive quality models.[15,67] On the basis of our results, we further argue that metrics are used because tools provide them rather than because of implementing a sound metric catalog or quality model designed to properly address the quality attributes of interest. We argue that this is also a reason for the obvious absence of metrics specific to automotive software development, because most metrics used in automotive software development projects are selected from such standard—often unadjusted—metric catalogs. The missing explicit links between metrics and quality attributes that have become obvious in the study at hand also support this statement. The various mappings and the links between metrics and quality attributes (including different standards) constitute a first step toward such an integrated perspective yet require further research. As a major outcome of our study, we created a large structured dataset that links together

- A set of 112 metrics collected from a systematic literature review (SLR),
- Two standards/metric catalogs used in the automotive industries (ISO/IEC 26262:2018 and HIS),
- A general standard on product quality (ISO/IEC 25010:2011, including ISO/IEC 25023:2016), and
- A set of 20 selected tools used to assess and monitor software quality.

Our findings lay the foundation for creating a knowledge base that helps practitioners select appropriate metric sets to measure quality attributes of interest. Instead of defining a new comprehensive quality model, which would however be the most straightforward solution, we propose using the different available information blocks and to combine them in a pragmatic way. That is, we propose to use all relevant standards for automotive software development and to provide mappings and clearly defined characteristics (as, for instance, described in the Quamoco approach[15]). Together with detailed information about the actual tool chain in a company or a project, a recommender system can compute proper metric sets that (i) provide the required coverage of standards and (ii) align the tools available or show gaps that need to be filled. Figure 13 illustrates the resulting model from a bird's-eye perspective. The figure illustrates how a respectively designed future knowledge system can be asked, for example, for quality attributes of interest and returns proper metric sets (including alternative metrics) and can also recommend tools that can handle the proposed metric sets. The bidirectional arrows mean that there is a relationship between the records about properties and IDs in the datasets. The unidirectional arrows describe a possible mapping into the dataset. All records can be linked together: for instance, by mapping HIS-metrics to the SLR-metrics, the ISO/IEC 25010:2011 and the supporting tools can be used to select those tools that support HIS-metrics while fulfilling the quality criteria of ISO/IEC 25023:2016. The connection can be established through the findings of our systematic literature review. The output generates a table with the corresponding information from ISO/IEC 25023:2016, ISO/IEC 25010:2011, and

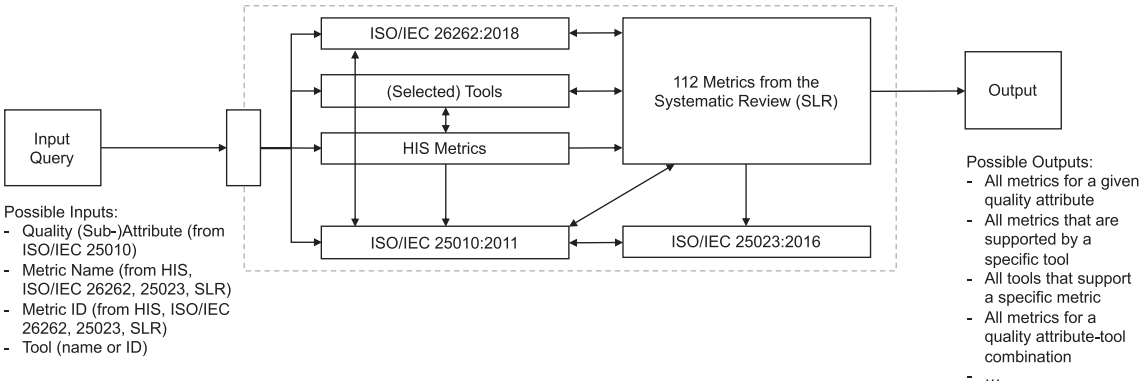


FIGURE 13 Overview of the result model and outline of future work

ISO/IEC 26262:2018. Metrics that fulfill all three ISO standards and that are supported by tools thus provide valuable information. However, as this study provides the data basis only, implementation and evaluation of such a knowledge system remains subject to future work, but recent research, for example, Tsuda et al.,[14] shows the relevance of flexible and more precise measurement and evaluation frameworks.

5.3 | Threats to validity

We discuss the threats to validity of our study following the categorization by Wohlin et al.[68] A general threat to be discussed is the publication bias, that is, the phenomenon that positive results of a study are more likely published than negative ones.[26,69] As a literature study, the study at hand is affected by this particular threat, for example, by suffering from potential incompleteness of the search results and the general publication bias. Furthermore, we decided to ground our study in the WoS meta-search engine only. The analyzed articles from the literature might not include publications reporting failed experiments and the lessons learned from such experiences, and, besides the handpicked papers, our result set does not contain papers published in venues not indexed by the WoS. Whereas the first issue cannot be resolved in the context of the study at hand, the second issue (meta-search engine) can be mitigated by other researchers reimplementing the research design described in Section 3 while considering the subsequently discussed specific threats to validity.

5.3.1 | Internal validity

The internal validity could be threatened by personal ratings of the researchers involved in the paper selection (selection bias). To address this risk, we followed a proven procedure[24-26] that, among other things, includes researcher triangulation to support dataset cleaning, study selection, study classification, and content analysis. For this, at least three researchers of the author team were not involved in actual analysis tasks but focused on the quality assurance only. The internal validity could also be affected by the limited data (only 38 papers resulting from the study selection). We mitigated this threat by applying a combined search strategy that includes manual selection, automated search, and a snowballing approach. Furthermore, information that was considered relevant in the study but was not available form the study's result set was collected by including external and “gray” literature, for example, the ECSS standard,[17] which was used to fill gaps in the dataset. Finally, the tools selected for the analysis of the applicability of the metrics have been selected in an opportunistic approach, which was influenced by the industrial context of the practitioners in the team. To mitigate this threat, the academic researchers initially provided a tool selection, which was evaluated and completed by the practitioners.

5.3.2 | External validity

The external validity is threatened by the missing knowledge about the generalizability of the results. Notably, the scope of the research could limit the generalizability as the field of application does not provide further information about the practical use of metrics. To mitigate this threat, we also analyzed papers at the borders of the field of interest, such as avionics, which also is a safety-critical industry sector. Furthermore, we also used the ECSS standard[17] as an external source of evidence complementing the general ISO-norms and thresholds. However, still, the findings of the study at hand need further independently conducted studies for cross-checking as our findings purely rely on literature, which suffers

from the selection bias.[26,69] Also, we only included peer-reviewed papers. That is, gray literature was excluded from the result set as well as PhD theses, white papers, and any other literature not fulfilling the quality assessment criteria outlined in Section 3.3.2. Gray literature was only included if gaps in the result set had to be closed, for example, by adding definitions or formulae.

5.3.3 | Conclusion validity

The conclusion validity might be impacted by the data included in the study. Specifically, the conclusions drawn from the systematic review could be too positive (publication bias) and grounded in an incomplete dataset (selection bias). To improve the conclusion validity, we included external standards, even outside the actual field of interest, into our study. Furthermore, conclusions drawn from the systematic review have been double-checked by researchers and practitioners not involved in the initial analysis.

6 | CONCLUSION

In this paper, we report findings from a systematic literature review in which we analyzed 38 papers for metrics used in automotive software development. Our study yielded 112 metrics of which 48 are supported by tools commonly used in automotive software development. A mapping of these metrics to the standards ISO/IEC 26262:2018 and ISO/IEC 25010:2011 and to the German HIS metric catalog showed a good coverage. The metrics defined in the HIS catalog and the SLR-metrics put emphasis on the *Maintainability* category of the ISO/IEC 25010:2011. Our analysis was focused on the quality attributes defined by the ISO/IEC 25010:2011; notably, we focused on the 31 subcategories to provide a proper categorization of the metrics. We developed a catalog of 112 metrics including a harmonized description/definition for these metrics, indication for tool support, and, where possible, formulae to illustrate how a metric is computed.

The findings from our study provide an extensive data basis to be used for the development of a knowledge-based support tool that will help practitioners select proper metrics in response to the quality attributes defined for a software product under consideration. Users of such a tool will be able to characterize a software product through the desired qualities. In response, users will be provided with a set of metrics properly addressing the quality attributes and a number of tools that support the measurement. Furthermore, users will be provided with "alternative" metrics; that is, if a specific metric is only supported by one tool, but another tool providing a better coverage of other metrics offers one or more substitutes, users can fine-tune their metric sets based on a tool configuration. However, to implement such a recommendation system, in the first step, further studies are required to fill those gaps that we identified in the study at hand. That is, our study revealed that several quality attributes are not (yet) well-covered by metrics found in our result set. Further studies with the purpose of identifying appropriate metrics are thus required. Moreover, the metric catalog developed in the study at hand needs to be checked with practitioners not involved in this study, notably regarding the practical relevance of a metric. The second step will include the development and the evaluation of a recommendation system as outlined above.

ORCID

Marco Kuhrmann https://orcid.org/0000-0001-6101-8931

REFERENCES

1. Manuel B. Mehr Software (im) Wagen: Informations- und Kommunikationstechnik (IKT) als Motor der Elektromobilität der Zukunft: Abschlussbericht des vom Bundesministerium für Wirtschaft und Technologie geförderten Verbundsvorhabens "eCar-IKT-Systemarchitektur für Elektromobilität". Online: https://doi.org/10.2314/GBV:728568616; 2010.
2. Manfred B. Software Eats the World Swiss Engineering Institute Press; 2015.
3. Ebert C, Kuhrmann M, Prikladnicki R. Global software engineering: an industry perspective. *IEEE Softw*. 2016;33(1):105-108.
4. ISO TC22/SC3/WG16. Road vehicles— Functional safety. ISO 26262:2018, International Organization for Standardization; 2018.
5. ISO/IEC 25010:2011. *Systems and software engineering – System and software quality models*: International Organization for Standardization; 2011.
6. Stefan W. *Software Product Quality Control*. Berlin Heidelberg: Springer; 2013.
7. *Systems and software engineering—vocabulary*: International Organization for Standardization; 2010.
8. Kan SH. *Metrics and Models in Software Quality Engineering*. 2nd ed.: Addison-Wesley Longman; 2002.
9. ISO/IEC 25023:2016. *Systems and software engineering—Systems and software Quality Requirements and Evaluation (SQuaRE)—measurement of system and software product quality*: International Organization for Standardization; 2016.
10. Sascha E-S, Nozomi Y-E, Klaus S. Metrics for analyzing variability and its implementation in software product lines: a systematic literature review. *Inform Softw Technol*. 2019;106:1-30.
11. Benjamin Treynor S, Shylaja N, Vivek R. Metrics that matter. *Queue*. 2018;16(6):30:86-30:105.
12. Wagner S, Lochmann K, Heinemann L, et al. The Quamoco product quality modelling and assessment approach. In: International Conference on Software Engineering; 2012:1-10.
13. Kläs M, Heidrich J, Münch J, Trendowicz A. CQML Scheme: A classification scheme for comprehensive quality model landscapes. In: 35th Euromicro Conference on Software Engineering and Advanced Applications; 2009:243-250.

14. Tsuda N, Washizaki H, Honda K, et al. WSQF: Comprehensive software quality evaluation framework and benchmark based on SQuaRE. In: International Conference on Software Engineering: Software Engineering in Practice; 2019:312-321.
15. Wagner S, Goeb A, Heinemann L, et al. Operationalised product quality models and assessment: the Quamoco approach. *Info Softw Technol*. 2015; 62(C):101-123.
16. Selvarani R, Gopalakrishnan Nair TR, Ramachandran M, Prasad K. Software metrics evaluation based on entropy. In: IGI Global; 2010:139-151.
17. ECSS Secretariat. Space product assurance, Software metrication programme definition and implementation. Standard ECSS-Q-HB-80-04A: ESAESTEC Requirements and Standards Division; 2011.
18. Prause CR, Werner J, Hornig K, Bosecker S, Kuhrmann M. Is 100% test coverage a reasonable requirement? Lessons learned from a space software project. *Product-Focused Softw Process Improv*. 2017;10611:351-367.
19. Antinyan V, Staron M, Sandberg A. Evaluating code complexity triggers, use of complexity measures and the influence of code complexity on maintenance time. *Empir Softw Eng*. 2017;22(6):3057-3087.
20. Hoffman RR, Hancock PA, Bradshaw JM. Metrics, metrics, metrics, Part 2: universal metrics? *IEEE Intell Syst*. 2010;25(6):93-97.
21. Alves TL, Ypma C, Visser J. Deriving metric thresholds from benchmark data. In: IEEE International Conference on Software Maintenance; 2010:1-10.
22. Schroeder J, Berger C, Staron M, Herpel T, Knauss A. Unveiling anomalies and their impact on software quality in model-based automotive software revisions with software metrics and domain experts. In: International Symposium on Software Testing and Analysis; 2016:154-164.
23. Schroeder J, Berger C, Knauss A, et al. Comparison of model size predictors in practice. In: International Conference on Software Engineering Companion (ICSE-C); 2017:186-188.
24. Petersen K, Feldt R, Mujtaba S, Mattsson M. Systematic mapping studies in software engineering. In: International Conference on Evaluation and Assessment in Software Engineering; 2008:68-77.
25. Kitchenham B, Budgen D, Brereton P. *Evidence-based software engineering and systematic reviews*: Apple Academic Press; 2015.
26. Kuhrmann M, Fernández D, Méndez DM. On the pragmatic design of literature studies in software engineering: An experience-based guideline. *Empir Softw Eng*. 2017;22(6):2852-2891.
27. Avizienis A, Laprie J-C, Randell B, Landwehr C. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans Dependable Secur Comput*. 2004;1(1):11-33.
28. Garousi V, Felderer M, Mäntylä MV. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Info Softw Technol*. 2019;106:101-121.
29. Badampudi D, Wohlin C, Petersen K. Experiences from using snowballing and database searches in systematic literature studies. In: International Conference on Evaluation and Assessment in Software Engineering; 2015:17:1-17:10.
30. Wieringa R, Maiden N, Mead N, Rolland C. Requirements engineering paper: a proposal and a discussion. *Require Eng*. 2005;11(1):102-107.
31. Shaw M. Writing good software engineering research papers. In: International Conference on Software Engineering; 2003:726-736.
32. Ivarsson M, Gorschek T. A method for evaluating rigor and industrial relevance of technology evaluations. *Empir Softw Eng*. 2011;16(3):365-395.
33. Durisic D, Nilsson M, Staron M, Hansson J. Measuring the impact of changes to the complexity and coupling properties of automotive software systems. *J Syst Softw*. 2013;86(5):1275-1293.
34. Gören S, Ferguson FJ. Test sequence generation for controller verification and test with high coverage. *ACM Trans Des Autom Electron Syst*. 2006; 11(4):916-938.
35. Mumtaz H, Alshayeb M, Mahmood S, Niazi M. An empirical study to improve software security through the application of code refactoring. *Info Softw Technol*. 2018;96:112-125.
36. Alshammari B, Fidge C, Corney D. Security metrics for object-oriented class designs. In: Ninth International Conference on Quality Software; 2009:11-20.
37. IEEE Std 1061-1992. IEEE standard for a software quality metrics methodology. IEEE; 1992.
38. Debou C, Haux M, Jungmayr S. A measurement framework for improving verification processes. *Softw Qual J*. 1995;4(3):207-225.
39. Pedrycz W, Han L, Peters JF, Ramanna S, Zhai R. Calibration of software quality: fuzzy neural and rough neural computing approaches. *Neurocomputing*. 2001;36(1-4):149-170.
40. Fenton Norman E, Neil M. Software metrics: successes, failures and new directions. *J Syst Softw*. 1999;47(2):149-157.
41. Yadav HB, Yadav DK. Construction of membership function for software metrics. *Proc Comput Sci*. 2015;46:933-940.
42. Huda S, Alyahya S, Ali M, et al. A framework for software defect prediction and metric selection. *IEEE Access*. 2018;6:2844-2858.
43. Altinger H, Dajsuren Y, Siegl S, Vinju JJ, Wotawa F. On error-class distribution in automotive model-based software. *Int Conf Softw Anal Evol Reeng*. 2016;1:688-692.
44. Komiyama T. Usability evaluation based on international standards for software quality evaluation. *NEC Tech J*. 2008;3(2):27-32.
45. Kandl S, Chandrashekar S. Reasonability of MC/DC for safety-relevant software implemented in programming languages with short-circuit evaluation. *Computing*. 2015;97(3):261-279.
46. Eclipse. Metrics Plugin Online: https://www.stateofflow.com/projects/16/eclipsemetrics; 2018.
47. MathWorks. Simulink Online: https://de.mathworks.com/help/slcheck/ref/model-metric-checks.html; 2018.
48. Blu Age Corporation. Blu Age Legacy Analysis Online: https://www.bluage.com/beam/media/bluage_legacy_analysis_en_20160126.pdf; 2016.
49. Acellere GmbH. GAMMA Online: https://help.mygamma.io/documentation/; 2018.
50. Parasoft. Parasoft Online: https://docs.parasoft.com/display/CPPDESKE1040/Metrics+Calculation; 2018.
51. M Squared Technologies LLC. Resource Standard Metrics (RSM) Online: https://www.msquaredtechnologies.com/base/Baseline-Diff-Metrics.html; 2018.
52. hello2morrow. Sonargraph Online: https://eclipse.hello2morrow.com/doc/standalone/content/index.html; 2018.
53. Designite. Designite for Java Online: https://www.designite-tools.com/DesigniteJava/; 2018.
54. Designite. Designite Online: https://www.designite-tools.com/; 2018.
55. FrontEndART Ltd. SourceMeter Online: https://www.sourcemeter.com/resources/java/; 2018.
56. FrontEndART Ltd. SourceMeter Online: https://www.sourcemeter.com/resources/cc/; 2018.
57. FrontEndART Ltd. SourceMeter Online: https://www.sourcemeter.com/resources/python/; 2018.
58. FrontEndART Ltd. SourceMeter Online: https://www.sourcemeter.com/resources/rpg/; 2018.
59. ZEN PROGRAM LTD. NDepend Online: https://www.ndepend.com/features/code-quality\#Metrics; 2018.
60. PMD Open Source Project. PMD Online: https://pmd.github.io/pmd-6.8.0/pmd_apex_metrics_index.html; 2018.

61. Baudin P. Frama-C. Online: https://frama-c.com/metrics.html; 2018.
62. Ristanovic CASE. Development Assistant for C (DAC) Online: https://www.ristancase.com/cms/v-dac-software-metrics-calculator; 2018.
63. Rouge Wave Software. Klocwork Online: https://docs.roguewave.com/en/klocwork/2018/metricsreference; 2018.
64. MathWorks. Polyspace Code Prover Online: https://de.mathworks.com/products/polyspace-code-prover.html; 2018.
65. QA Systems GmbH. QA-C Online: https://www.qa-systems.de/tools/qa-c/static-analysis/; 2018.
66. Mordal K, Anquetil N, Laval J, Serebrenik A, Vasilescu B, Ducasse S. Software quality metrics aggregation in industry. *J Softw Evol Process*. 2013; 25(10):1117-1135.
67. Mordal-Manet K, Balmas F, Denier S, et al. The squale model—a practice-based industrial quality model. In: IEEE International Conference on Software Maintenance; 2009:531-534.
68. Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A. *Experimentation in Software Engineering*. Berlin Heidelberg: Springer-Verlag; 2012.
69. Song F, Parekh S, Hooper L, et al. Dissemination and publication of research findings: an updated review of related biases. *Health Technol Assessment*. 2010;14(8):iii-193. https://doi.org/10.3310/hta14080
70. Luo Y, Stelma J, Brand M. Functional safety measurement in the automotive domain: adaptation of PSM. In: First International Workshop on Automotive Software Architecture; 2015:11-17.
71. Nassar B, Scandariato R. Traceability metrics as early predictors of software defects? In: IEEE International Conference on Software Architecture; 2017:235-238.
72. Garg M, Lai R. Measuring the constraint complexity of automotive embedded software systems. In: International Conference on Data and Software Engineering; 2014:1-6.
73. Kruger IH. Service-oriented software and systems engineering—a vision for the automotive domain. In: ACM/IEEE International Conference on Formal Methods and Models for Co-Design; 2005; Verona, Italy:150.
74. Staron M, Meding W, Karlsson G, Nilsson C. Developing measurement systems: an industrial case study. *J Softw Mainten Evol Res Pract*. 2011;23(2): 89-107.
75. Ince DC. Software metrics: introduction. *Info Softw Technol*. 1990;32(4):297-303.
76. Pfleeger SL. Software metrics: progress after 25 years? *IEEE Softw*. 2008;25(6):32-34.
77. Venkitachalam H, Richenhagen J, Schlosser A, Tasky T. Metrics for verification and validation of architecture in powertrain software development. In: First International Workshop on Automotive Software Architecture; 2015:27-33.
78. Abdallah A., Feron E., Hellestrand G., Koopman P., Wolf M. Hardware/software codesign of aerospace and automotive systems. *Proc IEEE*. 2010; 98(4):584-602.
79. Pelosi G, Barenghi A, Maggi M, Agosta G. Design space extension for secure implementation of block ciphers. *IET Comput Dig Techniq*. 2014;8(6): 256-263.
80. Balboni A, Fornaciari W, Ssciuto D. Partitioning of hardware–software embedded systems: a metrics-based approach. *Integ Comput-Aided Eng*. 1998; 5(1):39-56.
81. Costello RJ, Liu D-B. Metrics for requirements engineering. *J Syst Softw*. 1995;29(1):39-63.
82. Gaver DP, Jacobs PA. Reliability growth by failure mode removal. *Reliab Eng Syst Safety*. 2014;130:27-32.
83. Hafnaoui I, Ayari R, Nicolescu G, Beltrame G. Scheduling real-time systems with cyclic dependence using data criticality. *Des Autom Embedded Syst*. 2017;21(2):117-136.
84. Jin X, Donze A, Deshmukh JV, Seshia SA. Mining requirements from closed-loop control models. *IEEE Trans Comput-Aided Des Integ Circuits Syst*. 2015;34(11):1704-1717.
85. Kiebusch S, Franczyk B, Speck A. Process-family-points. *Softw Process Workshop*. 2006;3966:314-321.
86. Lengauer C, Bougé L, Trystram D, et al. Parallelizing highly complex engine management systems. *Concurrency Comput Pract Exper*. 2017;29(15): e4115.
87. Kenneth L, Rogardt H. A practical approach to size estimation of embedded software components. *IEEE Trans Softw Eng*. 2012;38(5):993-1007.
88. Luo Y, van den Brand M. Metrics design for safety assessment. *Info Softw Technol*. 2016;73:151-163.
89. Lytz R. Software metrics for the Boeing 777: a case study. *Softw Qual J*. 1995;4(1):1-13.
90. Rajagopalan J, Srivastava PK. Introduction of a new metric "Project Health Index" (PHI) to successfully manage IT projects. *J Organiz Change Manag*. 2018;31(2):385-409.
91. Rook J, Medhat S. Using metrics to monitor concurrent product development. *Indust Manag Data Syst*. 1996;96(1):3-7.
92. Singh G, Gobrogge S. Information system architecture for developing reusable testplans for embedded software. *Microprocess Microsyst*. 2001;24(9): 453-461.
93. Wagner J, Keane JF. A strategy to verify chassis controller software-dynamics, hardware, and automation. *IEEE Trans Syst Man Cybern - Part A: Syst Humans*. 1997;27(4):480-493.
94. Wan J, Canedo A, Faruque MAA. Cyber-physical codesign at the functional level for multidomain automotive systems. *IEEE Syst J*. 2017;11(4):1-11.
95. Williams Byron J, Carver JC. Examination of the software architecture change characterization scheme using three empirical studies. *Empirical Softw Eng*. 2014;19(3):419-464.
96. Zhang X, Teng X, Pham H. Considering fault removal efficiency in software reliability assessment. *IEEE Trans Syst Man Cybern Part A: Syst Humans*. 2003;33(1):114-120.
97. Balboni A, Cefriel WF, Sciuto D. TOSCA: TOols for System Co-design Automation. Online: www.cefriel.it/eda/projects/tosca/tosca.htm; 1996.

How to cite this article: Vogel M, Knapik P, Cohrs M, et al. Metrics in automotive software development: A systematic literature review. *J Softw Evol Proc*. 2021;33:e2296. https://doi.org/10.1002/smr.2296

APPENDIX A: DATA STRUCTURES FOR DATA COLLECTION AND EXTRACTION

This appendix shows the data structures used for extracting the information in the systematic review as described in Section 3.3. The data structures were used for the first two phases shown in Figure 2, that is, the systematic mapping study for scoping the research as well as the actual systematic literature review for collecting, structuring, and analyzing the data. Table A1 shows all metadata that was collected for every paper included in the analysis. Besides collecting the metadata of the respective papers, we applied three standard classification schemas to characterize the dataset. Although we fully applied the rigor-relevance model as defined by Ivarsson and Gorschek,[32] the two classification schemas RTF and CTF were tailored for application in the study. Table A2 lists the categories used for the application of the RTF schema, and Table A3 shows the categories for the application of the CTF schema.

TABLE A1 Metadata collected for each paper

Field	Description
No.	Running number of the paper
Title	Title of the paper
Authors	Author list of the paper
Year	Year of publication
Keywords/tags	List of keywords/tags as provided by the authors and publishers
Abstract	The full abstract of the paper (if available)
Venue	The conference, journal, etc. of the paper
DOI	The paper's Digital Object Identifier (DOI)
Database	The database that generated the item in the paper list (source for manually selected paper or WoS)
General	Paper exclusion, voting model for the evaluation of the exclusion criteria (Table 2)
General	Paper inclusion, voting model for the evaluation of the inclusion criteria (Table 2)
RTF	Evaluation of a paper according to the research type facets according to Wieringa et al.[30]
CTF	Evaluation of a paper according to the contribution type facets according to Shaw[31]
Rigor/relevance	Evaluation of a paper according to the rigor/relevance model according to Ivarsson and Gorschek[32]

TABLE A2 Applied research type facets as proposed by Wieringa et al.[30]

Criteria	Description
Evaluation research	Implemented in practice, evaluation of implementation conducted; requires more than just one demonstrating case study
Solution proposal	Solution for a problem is proposed, benefits/application is demonstrated by example, experiments, or student labs; also includes proposals complemented by one demonstrating case study for which no long-term evaluation/dissemination plan is obvious
Philosophical paper	New way of thinking, structuring a field in form of a taxonomy or a framework, secondary studies like SLR[25] or SMS[24]
Opinion paper	Personal opinion, not grounded in related work and research methodology
Experience paper	Personal experience, how are things done in practice

TABLE A3 Applied contribution type facets, adopted from Shaw[31]

Criteria	Description
Model	Representation of observed reality by concepts after conceptualization
Theory	Construct of cause–effect relationships
Framework	Frameworks/methods related to SPI
Guideline	List of advices
Lessons	learned set of outcomes from obtained results
Advice	Recommendation (from opinion)
Tool	A tool to support SPI

TABLE A4 Overview of the data extracted for each metric found in the dataset

Field	Description
PaperID	Paper ID (field No. from Table A1) to identify the paper that contains a specific metric
MetricID	Unique ID of a metric (used to harmonize the metric sets)
Name (Synonym)	Name of a metric. In case of multiple occurrences or different naming, synonyms are also collected
Description (Paper)	The actual description of a metric as provided by the respective paper
Description (TUC)	After the harmonization, this field contains a harmonized description of a specific metric, which also includes the particularities of the different sources. This harmonization is done by the researchers from the Clausthal University of Technology (TUC) and aims at providing *one* agreed description of a metric
Formula Available	An indicator to evaluate if a formula was provided by the source that introduced a metric or if the formula was obtained from other sources in case the paper from the result set does not provide a formula
Formula	The formula used to compute the metric (if available)
Metric Type	The type of the metric, e.g., code metric or process metric
Artifact	The artifact that is addressed by the metric, e.g., code or models
Language	In case the metric is a code metric, this field lists the languages for which the metric is available
Assignment	Indicates whether the assignment of a specific metric to one or more of the 31 sub-categories has been made by the authors of a paper (value: 1) or if the assignment was made in the course of this study (value: 0)
ISO/IEC 25010	This group of Boolean fields assigns a metric to the eight top-level and, if possible, to the 31 subcategories of the ISO/IEC 25010:2011 Product Quality Model
Relevance 25010	This field scores how the metric fulfills the selected criteria from the ISO/IEC 25010:2011 in a classification 1 to 05: 1. The metric fulfills the criterion completely. 2. The metric fulfills the criterion partially (mostly if a criterion is assigned to multiple top-level and sub-categories). 3. The metric fulfills the criterion conditionally (e.g., if a measure provides results that required expert knowledge for interpretation). 4. The metric does not really fulfill the criterion (e.g., if a measure provides results that require further results provided by other metrics to come up with an interpretation). 5. The metric does not fulfill the criterion (in this case, the assignment has to be re-checked).
Relevance Practice	Evaluation of the (practical) use reported for a metric in a classification 1 to 5: 1. The metric is used in real-world systems, i.e., the metric is established and considered a standard metric. 2. The metric is reported to be used in industry projects (without further context information). 3. The metric is used in field studies, e.g., studies on open source software, GitHub mining studies. 4. The metric is used in lab environments, e.g., in research projects. 5. The metric is used in student projects/toy examples only, e.g., a Java project with only five classes.
Threshold	If available, boundary values (thresholds), i.e., limits for a metric and its specific application context are collected
Value Ranges	If available, value ranges for a metric and its specific application context are collected
Value Type	If available, the type of the values is collected, i.e., discrete or continuous
Tool Support	If available, if the metric is supported by a tool, this field indicates the tool support

For the data extraction, the data structure shown in Table A4 was used. As outlined in Section 3.3.2, the data structure was initially developed, but we evolved the data structure in the course of conducting the study. For instance, the field *Tool Support* was added when we agreed in the team that tools should be included in the study to improve the understanding of practical relevance (data field *Relevance Practice*). Table A4 also includes explanations for value ranges, classification categories, and grades.

Of special importance was the evaluation of the relevance of specific metrics in the context of the different standards and the relevance to practice. The different mappings and relevance ratings—as agreed in the study team—are in detail: ISO/IEC 25010. This mapping describes which of the 31 subcategories of the ISO/IEC 25010:2011 (Figure 1) are covered by a specific metric. Each metric was analyzed for its application to the elements needed for the calculation. Additionally, in workshops with the industrial partner, we discussed which quality criteria are met by a specific metric and which further quality criteria a metric might also fulfill. The assignment was made to the subcategories on the basis of the classification from the literature, practical experience from projects of the past, and current best practices. We also discussed if an evaluation of a fulfillment of the top-level categories was necessary. However, we decided to remain on the level of subcategories as, if necessary, the fulfillment of the top-level categories can be computed from the respective degree of fulfillment of the subcategories.

Relevance 25010. This rating describes to what extent a metric helps fulfill a subcategory. For this, the descriptions of the individual subcategories were analyzed in terms of their satisfiability. This satisfiability was discussed and graded in workshops with our industry partner and using

literature. Notably, for completed and ongoing projects, we discussed to which extent a specific metric can help fulfill a subcategory. On the basis of all these aspects, a metric was "graded" on the scale shown in Table A4.

Relevance Practice. This rating describes how widespread and therefore practically relevant a metric is. For this, in the workshops with the industry partners, we interviewed the industry partners and discussed their experience. Notably, we discussed which metrics were already known and even in use in the practitioner's day-to-day business. To get more input, it was agreed that tools be added to the study. Hence, a search for tools that support data collected and data analysis to provide metrics for software development projects was conducted (Section 4.5). The results of this extra search were used in addition to the main analyses, that is, the number of tool-supported SLR-metrics, to assess the (perceived) practical relevance.

APPENDIX B: RESULT SETS FROM THE SYSTEMATIC REVIEW

This appendix contains the list of papers obtained in the different stages of the paper selection process as outlined in Section 3.3, that is, the papers obtained in the first two phases shown in Figure 2. Table B1 lists the 12 reference papers selected during the manual search process, and Table B2 lists the 26 finally selected papers obtained from the automatic search.

TABLE B1 Overview of the 12 reference papers selected in the manual search process

Reference	Paper title
70	Functional safety measurement in the automotive domain: adaptation of PSM
71	Traceability metrics as early predictors of software defects?
72	Measuring the constraint complexity of automotive embedded software systems
73	Service-oriented software and systems engineering—a vision for the automotive domain
74	Developing measurement systems: an industrial case study
41	Construction of membership function for software metrics
75	Software metrics: introduction
40	Software metrics: successes, failures and new directions
76	Software metrics: Progress after 25 years?
42	A framework for software defect prediction and metric selection
43	On error-class distribution in automotive model-based software
77	Metrics for verification and validation of architecture in powertrain software development

TABLE B2 Overview of the 26 selected papers from the automated search process

Reference	Paper title
78	Hardware/software codesign of aerospace and automotive systems
79	Design space extension for secure implementation of block ciphers
80	Partitioning of hardware–software embedded systems: a metrics-based approach
81	Metrics for requirements engineering
38	A measurement framework for improving verification processes
33	Measuring the impact of changes to the complexity and coupling properties of automotive software systems
82	Reliability growth by failure mode removal
34	Test sequence generation for controller verification and test with high coverage
83	Scheduling real-time systems with cyclic dependence using data criticality
84	Mining requirements from closed-loop control models
45	Reasonability of MC/DC for safety-relevant software implemented in programming languages with short-circuit evaluation
85	Process-family-points
86	Parallelizing highly complex engine management systems
87	A practical approach to size estimation of embedded software components

TABLE B2 (Continued)

Reference	Paper title
88	Metrics design for safety assessment
89	Software metrics for the Boeing 777: a case study
35	An empirical study to improve software security through the application of code refactoring
39	Calibration of software quality: fuzzy neural and rough neural computing approaches
90	Introduction of a new metric "Project Health Index" (PHI) to successfully manage IT projects
91	Using metrics to monitor concurrent product development
92	Information system architecture for developing reusable testplans for embedded software
44	Usability evaluation based on international standards for software quality evaluation
93	A strategy to verify chassis controller software-dynamics, hardware, and automation
94	Cyber-physical codesign at the functional level for multidomain automotive systems
95	Examination of the software architecture change characterization scheme using three empirical studies
96	Considering fault removal efficiency in software reliability assessment

APPENDIX C: THE HIS METRIC CATALOG, ISO/IEC 25010:2011, 25023:2016, AND 26262:2018 METRICS AND MAPPINGS

In this appendix, we present the detailed mappings performed following the procedures described in Section 3.4 and in Appendix A1. At first, Table C1 and the following ones provide the detailed mappings performed in the course of studying RQ2 (Section 4.3).

The next public standard of relevance in the studied domain is the ISO/IEC 26262:2018. For the different development phases defined for road vehicles, ISO/IEC 26262:2018 provides 85 metrics. Table C3 provides a summary of these metrics and links these metrics to the ones found in the systematic review (including references to the papers that mention these metrics). Table C3 provides the input for the mapping of metrics to quality attributes as presented in Section 4.

As a baseline to analyze the different quality attributes, we opted for the ISO/IEC 25010:2011 (see also Sections 2.1 and 3.4). For this standard, the ISO/IEC 25023:2016 already provides a mapping of selected metrics to the quality attributes and also to the subcharacteristics. Figure C1provides an overview of this mapping with the quality attributes and subcharacteristics in the columns and the assigned metrics in the rows. Table C4 provides the mapping of the metrics defined in the ISO/IEC 25023:2016 with the metrics found in the systematic literature review. Finally, Figure C2 provides the mapping between the ISO/IEC 25010:2011 and the ISO/IEC 26262:2018.

TABLE C1 Mapping of the metrics extracted from the systematic review to the metrics provided by HIS metric catalog (the column "MetricID" refers to the unique ID of a metric used in Table D1)

HIS-Index	Name (HIS)	Abbreviation	Paper reference	MetricID	QTY
1	Density of comments	COMF	39-42	4	4
2	Number of paths	PATH	42	112	1
3	Number of jumps	GOTO	41	14	1
4	Cyclomatic complexity	v(G)	38-42	1	5
5	Number of calling functions	CALLING	33,35	3, 35	2
6	Number of called functions	CALLS	38,39	3, 11	2
7	Number function parameters	PARAM	44,77	109, 74	2
8	Number of instructions per functions	STMT	39-41,77	4	4
9	Number all call levels	LEVEL	38,39	10, 11	2
10	Number of return points	RETURN	41	14	1
11	Stability index	Si	45	4	1
12	Language complexity	VOC	39,41,42	4	3
13	Number of MISRA HIS Subset violations	NOMV	40	78	1
14	Number of MISRA violations per rule	NOMVPR	40	78	1
15	Number of recursions	ap_cg_cycle	–	–	0

TABLE C2 Mapping of the metrics found in the systematic review to ISO/IEC 25010:2011 quality attributes; the table also shows if a metric was assigned to a category by the authors of the respective paper or if it was done manually in the course of this study

ISO/IEC 25010:2011	In paper	Manual	MetricID, see Table D1
Functional appropriateness	–	40,70,74,80,88,90	6, 7, 8, 9, 16, 31, 64, 65, 66, 67, 72, 73, 83, 84, 86, 87, 88, 89, 90, 94, 97, 106
Functional correctness	–	40,70,84,88	6, 7, 8, 9, 64, 100
Functional completeness	–	88	83, 90
Availability	–	39,82,88,94	7, 33, 54, 103
Fault tolerance	–	34,39,94	34, 54, 60
Recoverability	–	39	54, 95
Maturity	45,71,80	38,39	48, 49, 54, 58, 80, 82, 91, 95, 107, 108
Time behavior	86	78,80,94	30, 42, 56, 57, 63, 68, 76, 79, 112
Resource utilization	86	42,80,94	15, 32, 38, 41, 48, 49, 62, 75, 77, 105
Capacity	70	80	6, 48, 49, 62, 105
Appropriateness recogn.	44	–	109
Learnability	44	–	53
Operability	–	–	–
User error protection	44	–	99
User interface aesthetics	44	–	69, 110
Accessibility	–	–	–
Confidentiality	–	35	17, 20, 21, 25
Integrity	–	35,88	7, 17, 18, 19, 20, 21, 22, 23, 24, 25, 28, 36, 43, 44, 45, 65, 66
Non-repudiation	–	35,45	18, 19, 20, 22, 24, 25, 36, 51, 55
Accountability	–	35,88	7, 51
Authenticity	–	35	19, 21
Co-existence	35	–	74
Interoperability	35	–	104
Modularity	–	33,35,38,39,42,45,72,77,80	1, 10, 11, 12, 17, 18, 19, 20, 24, 22, 26, 27, 29, 35, 36, 37, 38, 39, 40, 44, 45, 46, 51, 52, 74, 80, 85, 95, 102, 111
Reusabillity	39,85	33,35,38,41,42,45,72,77,80,83	1, 3, 5, 10, 11, 12, 20, 21, 22, 24, 25, 26, 27, 29, 35, 37, 44, 45, 47, 51, 52, 58, 74, 80, 81, 85, 93, 85, 101, 111
Analyzability	39	33,35,38-43,45,71,72,80,84	1, 2, 3, 4, 10, 11, 12, 14, 18, 24, 26, 28, 29, 35, 36, 37, 38, 39, 40, 43, 44, 45, 46, 50, 51, 52, 54, 55, 61, 63, 70, 71, 74, 80, 82, 85, 91, 95, 96, 98, 104, 107, 108, 111
Modifyability	–	33,35,38,39,42,45,72,83	3, 5, 10, 11, 12, 18, 19, 20, 22, 26, 27, 29, 35, 37, 38, 51, 52, 74, 80, 85, 95, 98, 111
Testability	77	33-35,38-40,42,45,83	1, 3, 5, 10, 11, 12, 17, 27, 35, 44, 45, 51, 52, 55, 30, 74, 78, 80, 82, 85, 95, 98, 104, 111
Replaceability	–	33,35,42	35, 37, 52, 98
Adaptability	–	33,35,77,80,84	12, 13, 27, 59, 61, 96, 106
Installability	–	80	106

TABLE C3 Identified metrics from the ISO/IEC 26262:2018 including a mapping to the metrics found in the systematic review (cf. Table D1) and the papers that refer to these metrics (the degree of recommendation to use corresponding methods depends on the ASIL: ++, method is highly recommended for the identified ASIL; +, method is recommended for the identified ASIL; o, method has no recommendation for or against usage for the identified ASIL; the ASIL integrity requirements are categorized into the categories A = lowest to D = highest)

			ASIL					
ID	Phase	Name	A	B	C	D	Paper	Metric ID
1	6.4.7	Suitability for software development						
2	6.4.7	Compliance and consistency with the technical safety requirements					[88]	87, 8, 9
3	6.4.7	Compliance with the system design					[44]	110
4	6.4.7	Consistency with the hardware-software interface						
5	7.4.1	Natural language	++	++	++	++		
6	7.4.1	Informal notations	++	++	+	+		
7	7.4.1	Semi-formal notations	+	++	++	++		
8	7.4.1	Formal notations	+	+	+	+		
9	7.4.3	Appropriate hierarchical structure of software components	++	++	++	++	[35]	85, 51
10	7.4.3	Restricted size and complexity of software components	++	++	++	++	[33,80]	81, 27, 26
11	7.4.3	Restricted size of interfaces	+	+	+	+	[77]	104, 74
12	7.4.3	Strong cohesion within each software component	+	++	++	++	[35]	12
13	7.4.3	Loose coupling between software components	+	++	++	++	[35,42]	35
14	7.4.3	Appropriate scheduling properties	++	++	++	++	[83]	5
15	7.4.3	Restricted use of interrupts	+	+	+	++	[45]	55
16	7.4.3	Appropriate spatial isolation of the software components	+	+	+	++		
17	7.4.3	Appropriate management of shared resources	++	++	++	++		
18	7.4.14	Walk-through of the design	++	+	o	o	[88]	97
19	7.4.14	Inspection of the design	+	++	++	++	[34]	60
20	7.4.14	Simulation of dynamic parts of the design	+	+	+	++		
21	7.4.14	Prototype generation	o	o	+	++		
22	7.4.14	Formal verification	o	o	+	+	[34]	60
23	7.4.14	Control flow analysis	+	+	++	++	[39]	10
24	7.4.14	Data flow analysis	+	+	++	++	[83]	5
25	7.4.14	Schedule analysis	+	+	++	++		
26	8.4.3	Natural language	++	++	++	++		
27	8.4.3	Informal notations	++	++	+	+		
28	8.4.3	Semi-formal notations	+	++	++	++		
29	8.4.3	Formal notations	+	+	+	+		
30	8.4.4	One entry and one exit point in sub-programs and functions	++	++	++	++	[38,77]	74, 1
31	8.4.4	No dynamic objects or variables, or else online test during their creation	+	++	++	++		
32	8.4.4	Initialization of variables	++	++	++	++	[80]	50
33	8.4.4	No multiple use of variable names	+	++	++	++	[39,41,42]	4
34	8.4.4	Avoid global variables or else justify their usage	+	+	++	++	[80]	50
35	8.4.4	Restricted use of pointers	o	+	+	++	[80]	50
36	8.4.4	No implicit type conversions	+	++	++	++		
37	8.4.4	No hidden data flow or control flow	+	++	++	++	[45]	55
38	8.4.4	No unconditional jumps	++	++	++	++	[41]	14
39	8.4.4	No recursions	+	+	++	++		
40	9.4.2	Walk-through	++	+	o	o	[88]	97
41	9.4.2	Pair-programming	+	+	+	+		
42	9.4.2	Inspection	+	++	++	++	[34]	60
43	9.4.2	Semi-formal verification	+	+	++	++		

(Continues)

TABLE C3 (Continued)

ID	Phase	Name	ASIL				Paper	Metric ID
			A	B	C	D		
44	9.4.2	Formal verification	o	o	+	+	[34]	60
45	9.4.2	Control flow analysis	+	+	++	++	[39]	10
46	9.4.2	Data flow analysis	+	+	++	++	[83]	5
47	9.4.2	Static code analysis	+	++	++	++	[38-42]	1
48	9.4.2	Static analyses based on abstract interpretation	+	+	+	+	[84]	59
49	9.4.2	Requirements-based test	++	++	++	++	[71,84]	96, 108
50	9.4.2	Interface test	++	++	++	++	[44]	110
51	9.4.2	Fault injection test	+	+	+	++		
52	9.4.2	Resource usage test	+	+	+	++		
53	9.4.2	Back-to-back comparison test between model and code, if applicable	+	+	++	++	[88]	97
54	9.4.3	Analysis of requirements	++	++	++	++	[88]	65
55	9.4.3	Generation and analysis of equivalence classes	+	++	++	++		
56	9.4.3	Analysis of boundary values	+	++	++	++		
57	9.4.3	Error guessing based on knowledge or experience	+	+	+	+	[45]	55
58	9.4.4	Statement coverage	++	++	+	+	[42]	38
59	9.4.4	Branch coverage	+	++	++	++	[41]	14
60	9.4.4	MC/DC (Modified Condition/Decision Coverage)	+	+	+	++	[45]	80
61	10.4.2	Requirements-based test	++	++	++	++	[71,84]	96, 108
62	10.4.2	Interface test	++	++	++	++	[44]	110
63	10.4.2	Fault injection test	+	+	+	++		
64	10.4.2	Resource usage evaluation	+	+	+	++		
65	10.4.2	Back-to-back comparison test between model and code, if applicable	+	+	++	++	[88]	97
66	10.4.2	Verification of the control flow and data flow	+	+	++	++		
67	10.4.2	Static code analysis	+	++	++	++	[38-42]	1
68	10.4.2	Static analyses based on abstract interpretation	+	+	+	+		
69	10.4.3	Analysis of requirements	++	++	++	++	[88]	65
70	10.4.3	Generation and analysis of equivalence classes	+	++	++	++		
71	10.4.3	Analysis of boundary values	+	++	++	++		
72	10.4.3	Error guessing based on knowledge or experience	+	+	+	+	[45]	55
73	10.4.5	Function coverage	+	+	++	++	[45]	82
74	10.4.5	Call coverage	+	+	++	++	[77]	13
75	11.4.1	Hardware-in-the-loop	+	+	++	++		
76	11.4.1	Electronic control unit network environments	++	++	++	++		
77	11.4.1	Vehicles	++	++	++	++		
78	11.4.2	Requirements-based test	++	++	++	++	[71,84]	96, 108
79	11.4.2	Fault injection test	+	+	+	++		
80	11.4.3	Analysis of requirements	++	++	++	++	[88]	65
81	11.4.3	Generation and analysis of equivalence classes	+	++	++	++		
82	11.4.3	Analysis of boundary values	+	++	++	++		
83	11.4.3	Error guessing based on knowledge or experience	+	+	+	+	[45]	55
84	11.4.3	Analysis of functional dependencies	+	+	++	++	[45]	82
85	11.4.3	Analysis of operational use cases	+	++	++	++		

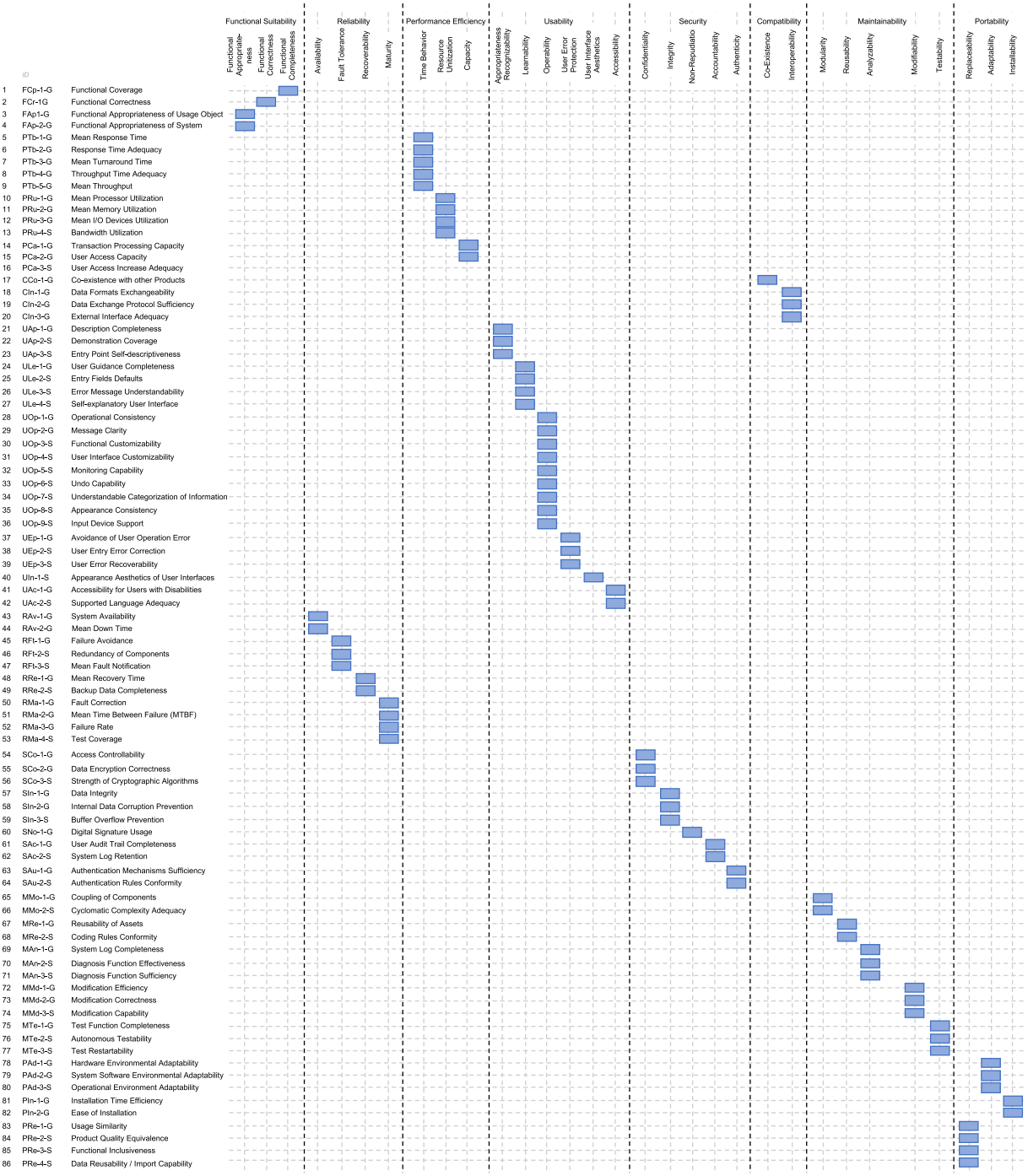


FIGURE C1 Mapping of quality attributes and metrics as provided by the ISO/IEC 25010:2011 and ISO/IEC 25023:2016

TABLE C4 Metrics from the ISO/IEC 25023:2016 including a mapping to the metrics found in the systematic review (cf. Table D1) and the papers that refer to these metrics

ID	ID ISO 25023	Name	Paper	Metric ID
2	FCr-1-G	Functional correctness	[45]	55
3	FAp-1-G	Functional appropriateness of usage objective	[88]	65, 83
4	FAp-2-G	Functional appropriateness of system	[88]	66, 84
5	PTb-1-G	Mean response time	[86]	76
6	PTb-2-G	Response time adequacy	[86]	56
7	PTb-3-G	Mean turnaround time	[80]	63
8	PTb-4-G	Turnaround time adequacy	[80]	57
9	PTb-5-G	Mean throughput	[86]	68
10	PRu-1-G	Mean processor utilization	[86]	42
11	PRu-2-G	Mean memory utilization	[86]	15
13	PRu-4-S	Bandwidth utilization	[78]	112
20	CIn-3-S	External interface adequacy	[77]	74, 104
21	UAp-1-G	Description completeness	[44]	110
22	UAp-2-S	Demonstration coverage	[44]	110
23	UAp-3-S	Entry point self-descriptiveness	[44]	69
24	ULe-1-G	User guidance completeness	[44]	110
26	ULe-3-S	Error message understandability	[44]	99
27	ULe-4-S	Self-explanatory user interface	[44]	53
43	RAv-1-G	System availability	[44,82]	54, 103
44	RAv-2-G	Mean down time	[82]	103
45	RFt-1-G	Failure avoidance	[34,88]	7, 60
46	RFt-2-S	Redundancy of components	[33]	52
47	RFt-3-S	Mean fault notification	[34]	60
48	RRe-1-G	Mean recovery time	[82]	103
50	RMa-1-G	Fault correction	[34]	60
51	RMa-2-G	Mean time between failure (MTBF)	[94]	33
52	RMa-3-G	Failure rate	[82]	103
53	RMa-4-S	Test coverage	[45]	80
54	SCo-1-G	Access controllability	[35]	17, 20, 21, 25
58	SIn-2-G	Internal data corruption prevention	[35]	20
61	SAc-1-G	User audit trail completeness	[88]	7
63	SAu-1-G	Authentication mechanisms sufficiency	[35]	19
65	MMo-1-G	Coupling of components	[33,42]	35, 37, 39, 40
66	MMo-2-S	Cycolmatic complexity adequacy	[38-42]	1, 95
67	MRe-1-G	Reusability of assets	[85]	93
69	MAn-1-G	System log completeness	[77]	74
70	MAn-2-S	Diagnosis function effectiveness	[42,77]	98, 104
71	MAn-3-S	Diagnosis function sufficiency	[39]	71
72	MMd-1-G	Modification efficiency	[33,35,42]	12, 19, 20, 22, 26, 25, 37, 51, 52, 85, 111
73	MMd-2-G	Modification correctness	[33,35,42]	12, 19, 20, 22, 26, 25, 37, 51, 52, 85, 111
74	MMd-3-S	Modification capability	[33,35,42]	12, 19, 20, 22, 26, 25, 37, 51, 52, 85, 111
75	MTe-1-G	Test function completeness	[34,45,77]	55,60, 74, 80, 82
76	MTe-2-S	Autonomous testability	[45,77]	80, 82, 104
78	PAd-1-G	Hardware environmental adaptability	[80]	106
79	PAd-2-G	System software environmental adaptability	[33,35,77]	12, 27, 92

(Continues)

TABLE C4 (Continued)

ID	ID ISO 25023	Name	Paper	Metric ID
80	PAd-3-S	Operational environment adapatability	84	96
84	PRe-2-S	Product quality equivalence	80	77
85	PRe-3-S	Functional inclusiveness	33	52
86	PRe-4-S	Data reusability / import capability	77	92

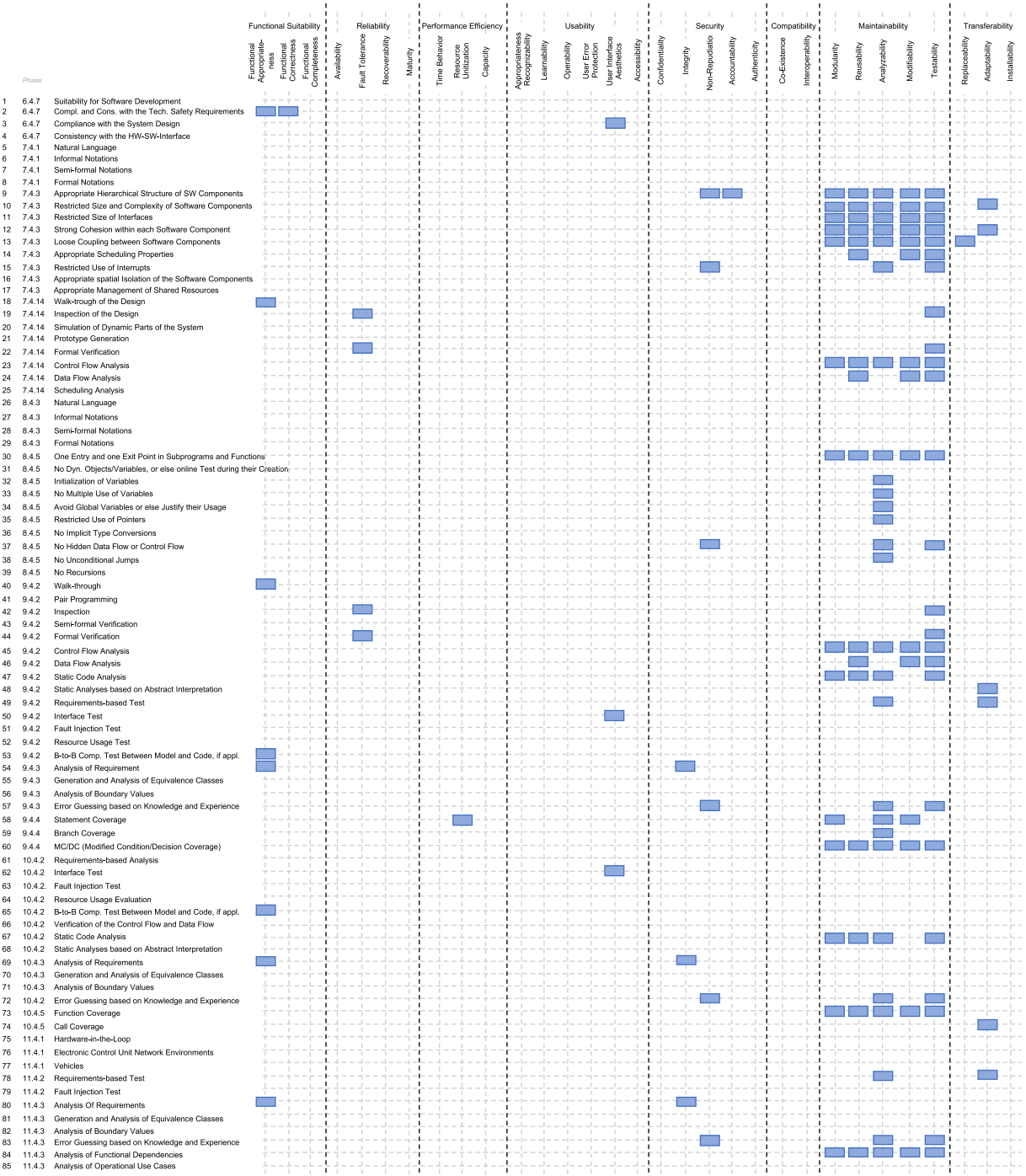


FIGURE C2 Mapping of quality attributes and metrics as provided by the ISO/IEC 25010:2011 and ISO/IEC 26262:2018

APPENDIX D: METRICS FROM THE SYSTEMATIC REVIEW

This appendix provides a compact overview of the metrics found in the systematic review. Table D1 provides this summary of the full dataset, which includes the metric's name, a short description, references to papers mentioning the metrics, values ranges (thresholds; where available) Table 5, equations, and the mapping to the ISO/IEC 25010:2011 quality attributes. The table also introduces the *Metric ID*, which is a unique identifier throughout the whole study; that is, whenever a metric is referred through its ID, this identifier refers to the unique metric ID presented in Table D1 (and also in the study's data).

TABLE D1 Overview of all metrics including full description, references, and formulae

ID	Metric Name	Description	References
1	Cyclomatic complexity number (McCabe)	*Description:* This metric provides an indication of code complexity based on the number of branches in control flows. *Forumla:* VG=#Edges−#Nodes+2 *ISO/IEC 25010 Category:* Reusability	38-42
2	Lines of Code (LoC)	*Description:* This is a very basic software metric, that includes the number of executable source instructions excluding comments, blank lines, and any non-executable lines. *Formula:* LoC=#CodeLines−#CmtsBlks *ISO/IEC 25010 Category:* Analyzability	38-42
3	Henry and Kafura (Fan In/Out)	*Description:* Fan-in represents the number of modules that are calling a given module, whereas fan-out represents the number of modules that are called by the given module. *Formula:* $C(i) = (f_{in}(i) \times f_{out}(i))^2$	33,39
4	Halstead Metrics	Halstead considers some primitive parameters of the implemented software product or program to measure its length, effort, development time, and other factors. *Formulae:* n_1 = the number of operators (distinct) n_2 = the number of operands (distinct) N_1 = the total number of operators N_2 = the total number of operands Program vocabulary: $n=n_1 \times n_2$ Program length: $N=N_1 \times N_2$ Estimated program length: $NO=n_1 log_2(n_1)+n_2 log_2(n_2)$ Program volume : $V=N \times log_2(n)$ Program difficulty: $D = \frac{n_1}{2} \times \frac{N_2}{n_2}$ Programming effort: $E=D \times V$ Estimated time: $T = \frac{E}{S}$	39,41,42
5	Error Propagation Approach (EPA)	*Description:* This metric measures the rate of error propagation of a scheduling real-time system. *Formula:* $\text{SysCrit} = \max(\text{CEP}(t_x))$	83
6	Actual Cost of Work Performed (ACWP)	*Description:* This metric measures the staff costs of the entire process. *ISO/IEC 25010 Category:* Capacity	70,88
7	Audit findings regarding ASIL	*Description:* This metric measures how many hazards are identified for each ASIL.	88
8	Audits on the details captured in functional requirements	*Description:* This metric is an indicator for all relevant details discussed for the functional safety requirements.	88
9	Audits on the relation between the ASIL of hazards and the derived safety goals	*Description:* This metric is an indicator if the associated ASIL of the functional safety requirements comply with the safety goals addressed.	88
10	Belady's bandwidth (nesting levels)	*Description:* This metric indicates the average level of nesting or width of the control flow graph representation of the program. *Formula:* $BW = \frac{1}{n} \sum_i i L_i$ *ISO/IEC 25010 Category:* Reusability *Threshold:* Table 5	39

(Continues)

TABLE D1 (Continued)

ID and Metric Name		Description	References
11	Depth of Nesting (DoN)	*Description:* This metric indicates the nesting level (NL) of a single routine (function or procedure). *Formula:* $NL = \max(\text{stmt}_{\text{nested}})$ According to[17], the number of nested statements includes simple or multiple-choice decisions, and loops in a routine. Then, the nesting level (X) of a module is defined as: $X = \max(NL)$ for all routines in the module. *Threshold:* Table 5	38
12	Lack of Cohesion in Methods (LCOM)	*Description:* The LCOM value provides a measure of the relative disparate nature of methods in the class. *Formula:* LCOM = number of disjoint subsets	35
13	Cohesion Number of requirements associated with a software component, Number of functions per software component, Average function interaction within components	*Description:* This metric provides an indication of the functions assigned to a single software module. *Formula:* Average function interaction = $\frac{\sum n_{iF}}{N_F}$	77
14	Branch count	*Description:* The branch count is virtually identical to "cyclomatic complexity". Branches are anything that jumps out of the current method.	41
15	Buffer Size (BufferSize)	*Description:* The BufferSize metric quantifies the additional required memory in bits needed to enforce data consistency by a buffering technique. *ISO/IEC 25010 Category:* Resource utilization	86
16	Certification costs	*Description:* This metric indicates what are the overhead costs for developing an item.	88
17	Classified Accessor Attribute Interactions (CAAI)	*Description:* This metric measures the interactions of accessors with classified attributes in a class. *Formula from*[84]: $\text{CAAI}(C) = \frac{\sum_{j=1}^{ca} \beta(\text{CA}_j)}{\lvert\text{AM}\rvert \times \lvert\text{CA}\rvert}$	35
18	Classified Attributes Inheritance (CAI)	*Description:* Classified Attributes Inheritance (CAI) is the ratio of the number of classified attributes that can be inherited in a hierarchy to the total number of classified attributes in this hierarchy. *Formula from*[84]: $\text{CAI}(H) = \frac{\lvert\text{AI}\rvert}{\lvert\text{CA}\rvert}$	35
19	Classified Attributes Interaction Weight (CAIW)	*Description:* This metric is defined to measure the interactions with classified attributes by all methods of a given class. *Formula from*[36]: $\text{CAIW}(C) = \frac{\sum_{j=1}^{ca} \gamma(\text{CA}_j)}{\sum_{i=1}^{a} \delta(A_i)}$	35
20	Classified Class Data Accessibility (CCDA)	*Description:* This metric measures the direct accessibility of classified class attributes of a particular class and aims to protect the classified internal representations of a class, i.e., class attributes, from direct access. *Formula from*[36]: $\text{CCDA}(C) = \frac{\lvert\text{CCPA}\rvert}{\lvert\text{CA}\rvert}$	35
21	Classified Instance Data Accessibility (CIDA)	*Description:* This metric measures the direct accessibility of classified instance attributes of a particular class and helps to protect the classified internal representations of a class, i.e., instance attributes, from direct access. *Formula from*[36]: $\text{CIDA}(C) = \frac{\lvert\text{CIPA}\rvert}{\lvert\text{CA}\rvert}$	35
22	Classified Methods Inheritance (CMI)	*Description:* This metric measures the ratio of the number of classified methods that can be inherited in a hierarchy to the total number of classified methods in that hierarchy. *Formula from*[84]: $\text{CMI}(H) = \frac{\lvert\text{MI}\rvert}{\lvert\text{CM}\rvert}$	35
23	Classified Methods Weight (CMW)	*Description:* This metric is defined to determine the weight of methods in a class that potentially interact with other methods. *Formula:* $\text{CMW}(H) = \frac{\lvert\text{CM}\rvert}{\lvert M\rvert}$	35
24	Classified Mutator Attribute Interactions (CMAI)	*Description:* This metric measures the interaction of mutators with classified attributes in a class. *Formula from*[36]: $\text{CMAI}(C) = \frac{\sum_{j=1}^{ca} \alpha(\text{CA}_j)}{\lvert\text{MM}\rvert \times \lvert\text{CA}\rvert}$	35

(Continues)

TABLE D1 (Continued)

ID and Metric Name		Description	References
25	Classified Operation Accessibility (COA)	*Description:* This metric measures the ratio of access possibilities to publicly classified methods of a particular class. *Formula from*[36]: $COA(C) = \frac{\lvert CPM \rvert}{\lvert CM \rvert}$	35
26	Single Component	*Description:* This metric based on the logic of fan-in and fan-out and measures input and output complexities of one software component based on complexity attributes such as hierarchical level. *Formula:* $C(i)=cin(i)\times cout(i)$ *Threshold:* Table 5	33
27	Component Complexity	*Description:* This metric measures the input and output complexity for each software component in the system. *Formula:* $Coup(P_l^a, P_l^b) = \sum_{i=1}^{n}\sum_{j=1}^{m} r(e_{l+1}^i, e_{l+1}^j) + \sum_{j=1}^{m}\sum_{i=1}^{n} r(e_{l-1}^j, e_{l-1}^i)$	33
28	Composite-Part Critical Classes (CPCC)	*Description:* The metric is the ratio of the number of critical component classes to the total number of critical classes. *Formula:* $CPCC(D) = 1 - \left(\frac{\lvert CP \rvert}{\lvert CC \rvert}\right)$	35
29	Constraint Complexity Measurement Method	*Description:* This metric count the occurrences of each of the group of standard OCL clause contributing to the constraint complexity in terms of usage. *Formula:* $V(GC_x) = \sum_{i=1}^{p}\sum_{j=1}^{p} V(GE_{ij})$	72
30	Cost of the architecture (performance)	*Description:* This metric measures based on the execution time of the control tasks on the specific controllers, and the reaction time of the physical process and calculates the total delay of each path in the graph of the architecture. *Formula:* $Cost_{perf}^{arch} = max\left(\sum_{c_i \in path_0} \tau_i, \sum_{c_i \in path_n} \tau_i\right)$	94
31	Cost of the architecture (perspective)	*Description:* Selection of a specific architecture for an automotive design depends on various nonfunctional requirements and/or associated design costs. Therefore, the synthesis algorithm needs to automatically validate the system from different perspectives. Each perspective is mapped to the user requirements. For any valid design, the cost from any perspective must not violate the corresponding requirement. *Formula:* $Cost_{perf}^{arch} = f_{pes}(C)$	94
32	Cost of the architecture (power)	*Description:* This metric is calculated using the power efficiency, defined as the total output power divided by the total input power. *Formula:* $Cost_{power}^{arch} = \frac{Power_{c_0}}{Power_{c_i}}$	94
33	Cost of the architecture (reliability)	*Description:* This metric measures the failure rate of each component in the architecture model (MTBF). *Formula:* $Cost_{rel}^{arch} = \prod_{c_i \in arch} (1 - FR_i(MTBF))$	94
34	Cost of the architecture (robustness)	*Description:* This metric indicates the quality of the whole architecture under different environments. *Formula:* $Cost_{robust}^{arch} = \frac{Cost_{rel}^{arch}(E_{harsh})}{Cost_{rel}^{arch}(E_{norm})}$	94
35	Coupling between objects (CBO)	*Description:* This metric measures the number of classes used by another class.	35,42
36	Critical Classes Coupling (CCC)	*Description:* The Critical Classes Coupling (CCC) metric aims to determine the degree of coupling between classes and classified attributes in a given design. *Formula:* $CCC(D) = \frac{\sum_{j=1}^{ca} \alpha(CA_j)}{(\lvert C \rvert - 1) \times \lvert CA \rvert}$	35
37	Data abstraction coupling (DAC)	*Description:* This metric is defined as the total number of other class types need as an attribute in a class. *Formula:* DAC=number of ADTs defined in a class	42
38	Message passing coupling (MPC)	*Description:* This metric is define as the total number of call statements identified in a class or the number of messages passed between objects within the local method of a class. *Formula:* MPC=number of send statements defined in a class	42

(Continues)

TABLE D1 (Continued)

ID and Metric Name		Description	References
39	Package Coupling Coup	*Description:* This metric calculates the strengths of dependencies between software components belonging to different sub-systems/domains.	33
40	Package Coupling Metrics (PCM)	*Description:* This metric calculates the number of dependencies between software components contained inside the packages. *Formula:* $\mathrm{PCM}(P_a^l) = \sum_{b=1 \wedge b \neq a}^{t} \mathrm{Coup}(P_a^l, P_b^l)$	33
41	Communication	Only mentioned in text without further description	80
42	CPU Load (CPULoad)	*Description:* The metric quantifies the average load of a processor or individual core over the complete time span covered by the simulation. *ISO/IEC 25010 Category:* Time Behavior	86
43	Critical Design Proportion (CDP)	*Description:* The metric simply takes into account the size, i.e., the number of classes, in a given program.	35
44	Critical Superclasses Inheritance (CSI)	*Description:* The metric is defined as the ratio of the sum of classes that can inherit from each critical superclass to the number of possible inheritances from all critical classes in a class hierarchy.	35
45	Critical Superclasses Proportion (CSP)	*Description:* This metric is the ratio of the number of critical superclasses to the total number of critical classes in an inheritance hierarchy.	35
46	Data Dependencies among Data Declarations	*Description:* This metric is part of the TOSCA environment.[97]	80
47	Data-DC Complexity	Only mentioned in text without further description	38
48	DataMem(B)	This metric calculates the space for data of a basic block by considering the memory occupation. *Formula:* $\mathrm{DataMem}(B) = \sum n \times \mathrm{Byte}(\mathrm{dec}-1)$ *ISO/IEC 25010 Category:* Maturity	80
49	DataMem(M)	This metric calculates the data space for a software module. *Formula:* $\mathrm{DataMem}(M) = \sum \mathrm{DataMem}(m-i) + \sum n \times \mathrm{Byte}(\mathrm{dec}-1)$ *ISO/IEC 25010 Category:* Maturity	80
50	Declarations Count	*Description:* This metric is part of the TOSCA environment.[97]	80
51	Depth of Inheritance Tree (DIT)	*Description:* This metric calculates the degree of complexity caused by the inheritance.	35
52	Directed Dependency between Software Components	*Description:* This metric measures the number of exchanged signals between software components and their type. *Formula:* $w(s,s^{`}) = \sum_{i=1}^{l} \mathrm{type}(sig_{s \rightarrow s'}(i))$	33
53	Ease of Function Learning	*Description:* This metric uses user tests to calculate how quickly a user can learn a program. *Formula:* T is the mean time to learn to use a function correctly; with $0<T$ meaning, the shorter the time to learn, the better *Threshold:* Table 5 *ISO/IEC 25010 Category:* Learnability	44
54	Entropy Measure	*Description:* The entropy measure is defined as information flows in a complex class, the inter-object complexity between objects and the program complexity ((class complexity, inter-object complexity).It is expected that the quality, maintainability and understandability will increase throughout lifetime of a system. *Formula:* $H(X) = -\sum_{i=1}^{N-1} p_i log_2 p_i$	39
55	Error-detection	*Description:* This metric calculates for a given set of M distinct tests, the probability of detecting an error in an incorrect implementation of a Boolean expression. *Formula:* $P_{(N,M)} = 1 - \left[\frac{2^{(2^N-M)}-1}{2^{2^N}}\right]$ *Threshold:* Table 5	45

(Continues)

TABLE D1 (Continued)

ID and Metric Name		Description	References
56	Event-Chain Duration (ECDuration)	*Description:* The metric quantifies the time span between a stimulus and response event of an Event-Chain. Thus, the reaction time of critical processing paths in the system, e.g., across multiple REs of different tasks can be evaluated. *ISO/IEC 25010 Category:* Time Behavior	86
57	Execution Time (TimeEx)	*Description:* This metric measures the execution time of a given macro node M having N child nodes. *Formula:* $\text{TimeEx}(M) = \sum \text{Freq}(\text{Child}_i) \times \text{TimeEx}(\text{Child}_i)$	80
58	Expression-EC	Only mentioned in text without further description	38
59	FALSIFYALGO Algorithm	*Description:* This metric measures weighted STL semantics that associate a weight with each predicate to normalize the numerical difference and improve the expressiveness. *Formula:* Metric is computed using a complex algorithm that can be obtained from.[84]	84
60	Fault Coverage	*Description:* This metric measures the number of mutation of an FSM implementation that is incompatible with the FSM specifications. *Formula:* Metric is computed using a complex algorithm that can be obtained from.[34]	34
61	FINDPARAM Algorithm	*Description:* This metric is an indicator for the falsification problem: instead of minimizing the satisfaction function in an attempt to make it negative, we can try to maximize it in an attempt to make it positive. *Formula:* Metric is computed using a complex algorithm that can be obtained from.[84]	84
62	FixThrMetric	*Description:* This metric is an indicator for additional performance with respect to the design requirements in data-intensive applications. *Formula:* $\text{FixThrMetric} = \frac{\text{power}}{\text{throughput}} \approx \frac{\text{energy}}{\text{operation}}$	80
63	Frequency Execution (Freq)	*Description:* This metric measures the number frequency of execution of the nodes. *Formula:* $\text{Freq}(N_j) = \sum_{i=1}^{\text{nodes}} \text{Freq}(N_i) \times \text{Prob}(e_{ij})$	80
64	Function Points (FP)	*Description:* Function points are used in software development as a basis for cost estimation, benchmarking and generally for the derivation of productivity and quality metrics. A function-point rating is independent of the underlying technology of the application.	40
65	Functional Safety Requirements and Safety Goals	*Description:* This metric measures how many functional safety requirements are derived.	88
66	Impacts met by the safety plan and its activities	*Description:* This metric is an indicator if all results from the impact analysis incorporated in the safety plan and it activities.	88
67	In-Service-Performance (ISP)	*Description:* This metric is an indicator for a current situation in a company to predict required reactive measurements systems to support changing metric programs.	74
68	Inter-Core Communication Rate (ICCrate)	*Description:* The metric quantifies the amount of data in bits per time unit, which is exchanged between the cores. It is an indicator for the expected cross-core communication overhead. *ISO/IEC 25010 Category:* Time Behavior	86
69	Interface Appearance Customizability	This metric is an indicator for user behavior. *Formula:* $X = \frac{A}{B}$ A denotes the number of interfaces and B the number of interface elements. For $0 \le X \le 1$ holds: the closer X is to 1.0, the better the result. *Threshold:* Table 5 *ISO/IEC 25010 Category:* User interface aesthetics	44

(Continues)

TABLE D1 (Continued)

ID and Metric Name		Description	References
70	Interfacing	Only mentioned in text without further description	80
71	Jensen's estimator of program length	*Description:* This metric is an extension of the Halstead estimator for real time application programs.	39
72	Legal costs	*Description:* This metric is an indicator for the overhead costs for developing an item.	88
73	Licensing costs	*Description:* This metric is an indicator for the overhead costs for developing an item.	88
74	Maintainability (average output interface size)	*Description:* This metric is an indicator for the average size of the component's output interface is a measure of the maintainability of the system for model-based development. *Formula:* Average output interface size of the component = $\frac{n_o}{N}$	77
75	Maximum Load Distance (MaxLoadDist)	*Description:* The *MaxLoadDist* metric quantifies to what extent the overall load is equally distributed to the individual cores. *ISO/IEC 25010 Category:* Resource Utilization	86
76	Maximum Normalized Response Time (mNRT)	*Description:* The *mNRT* metric quantifies the relative worst-case response time that occurred in a simulation. *ISO/IEC 25010 Category:* Time Behavior	86
77	MaxThrMetric	*Description:* This metric is an indicator for the maximum throughput case, where the main goal is to maximize the speed, as typically happens in real-time systems. *Formula:* MaxThrMetric = $\frac{\text{energy}_{MAX}}{\text{throughput}_{MAX}} \approx \frac{\text{power}}{\text{throughput}^2}$	80
78	Metrics relating to defect counts	Only mentioned in text without further description	40
79	Microstate Count(MS)	*Description:* This metric count the number of micro-states in an time multiplexing model. *Formula:* $MS = MAX_{op}\left(\left\lceil \frac{n_{op}}{d_{op}} \right\rceil\right) ClockDelay_{op}$	80
80	Modified Condition/ Decision Coverage (MC/DC)	This metric is used to ensure adequate testing of critical software. *ISO/IEC 25010 Category:* Maturity	45
81	Module Use	*Description:* This metric is part of the TOSCA environment.[97]	80
82	Multiple condition coverage (MCC)	*Description:* This metric is used to ensure adequate testing of critical software. *ISO/IEC 25010 Category:* Maturity	45
83	Number of audit findings regarding hazards	*Description:* This metric is an indicator if the identified hazards complete.	88
84	Number of audit findings regarding process	*Description:* This metric is an indicator how well does the work comply with the standard requirements on the process.	88
85	Number of Children (NOC)	*Description:* This metric is an indicator for assessment the degree of complexity caused by inheritance.	35
86	Number of documents requiring rework	*Description:* This metric is an indicator for how much rework is required for redefining an item.	88
87	Number of functional safety requirements	*Description:* This metric count how many functional safety requirements are derived.	88
88	Number of hazards per ASIL	*Description:* This metric count how many safety goals are identified for each ASIL.	88
89	Number of hazards without a safety goal	*Description:* This metric is an indicator if all hazards transformed to safety goals.	88
90	Number of safety goals not covered by requirements	*Description:* This metric is an indicator if all safety goals covered by the functional safety requirements.	88
91	Number of Statements (NS)	Only mentioned in text without further description	38
92	Portability Dependency on basic software	*Description:* This metric is an indicator to determine the dependency of interfaces of the basic software on third-party software and the effects of a change in the underlying basic software platform. *Formula:* Dependency on basic software = $\frac{n_{iBSW.}}{n_i} \times 100$	77

(Continues)

TABLE D1 (Continued)

ID and Metric Name		Description	References
93	Process-Family-Points (PFP)	*Description:* This metric is an indicator for software system families to support a structured reuse of components and a high degree of automation based on a common infrastructure. *ISO/IEC 25010 Category:* Reusability	85
94	Project Health Index (PHI)	*Description:* This metric is an indicator for the relative importance of the various project management factors. *Formula:* Metric is computed using a experienced-based equation that can be obtained from.[90]	90
95	Reachability Measure	*Description:* This metric is an extension of McCabe and Halstead by Cobbs and measures graph-oriented length- and width-type that could defined one could formulate a meaning of "area", as a function of the length- and width-type measures. *ISO/IEC 25010 Category:* Reusability	39
96	Requirement Mining Algorithm (STL, PSTL)	*Description:* The algorithm determines requirements from closed-loop models with the help of a requirements template expressed in parametric signal temporal logic: a logical formula in which concrete signal or time values are replaced with parameters. *Formula:* Metric is computed using a complex algorithm that can be obtained from.[84]	84
97	Requirements met by Scenarios	*Description:* This metric is an indicator for how well does the work comply with the standard requirements on the process.	88
98	Response for a Class (RFC)	*Description:* This metric count the number of all possible methods to be executed. It evaluates all possible direct and indirect method calls that can be reached via associations. *Formula:* RFC of a class is defined as the sum of the number of methods in the class and the number of external methods directly called by those methods. *Threshold:* Table 5	35,42
99	Self-explanatory Error Messages	*Description:* This metric is an indicator for observe user behavior. *Formula:* $X = \frac{A}{B}$ A denotes the number of error conditions and B the number of error conditions tested. For $0 \le X \le 1$ holds: the closer X is to 1.0, the better the result. *Threshold:* Table 5 *ISO/IEC 25010 Category:* User error protection	44
100	Signal Temporal Logic	*Description:* This metric is an indicator for the temporal behaviors of reactive systems; originally input-output systems with Boolean and discrete-time signals. *Formula:* Metric is computed using a complex algorithm that can be obtained from.[84]	84
101	Statement (SC)	Only mentioned in text without further description	38
102	Structure of Module	*Description:* This metric is part of the TOSCA environment.[97]	80
103	System Reliability Growth	*Description:* This metric is an indicator to assess system reliability growth. *Formula:* Metric is computed using a complex algorithm that can be obtained from.[82]	82
104	Testability (average input interface size)	*Description:* This metric is an indicator for how can the testability of an individual software component be determined at the architecture level based on the number of interfaces. *Formula:* Average input interface size of the component = $\frac{n_i}{N}$ *ISO/IEC 25010 Category:* Testability	77

(Continues)

TABLE D1 (Continued)

ID and Metric Name		Description	References
105	Throughput	*Description:* This metric is an indicator for design purposes by analyze the system at the macro-node level, by considering its through-put depending on the number of operations to be completed and the delay under the general assumption of resources limitation. *Formula:* throughput = $\frac{\text{operations}}{\text{second}} \approx \frac{\text{concurrency}}{\text{delay}}$	80
106	Total Area	*Description:* This metric is an indicator for the space of a target architecture on a silicon area. *Formula:* $\text{Total Area} = A_{\text{CPU}} + A_{\text{memO.S.}} + A_{\text{mem(data,sw)}} + \sum_{i=1}^{n}(A_{\text{I/O}} + A_{\text{datapath}} + A_{\text{mem}} + A_{\text{control}})$	80
107	Traced Components per Requirement (CR)	*Description:* This metric measures how many components are traced to a requirement. *Formula:* $C^R = \forall R_i : \sum_{j=1}^{\text{comp}} \text{IsLinked}(C_j, R_i)$ *ISO/IEC 25010 Category:* Maturity	71
108	Traced Requirements per Component (RC)	*Description:* This metric measures how many requirements are traced to a component. *Formula:* $R^C = \forall C_i : \sum_{j=1}^{\text{req}} \text{IsLinked}(C_i, R_j)$ *ISO/IEC 25010 Category:* Maturity	71
109	Understandable input and output	*Description:* This metric is an indicator for the number of input and output items understood by a user. *Formula:* $X = \frac{A}{B}$ *A* denotes the number of input and output data and *B* the number of input and output available from the interface. For $0 \le X \le 1$ holds: the closer *X* is to 1.0, the better the result. *Threshold:* Table 5 *ISO/IEC 25010 Category:* Appropriateness Recognizability	44
110	Usability Compliance	*Description:* This metric is an indicator for the specify required compliance items based on standards, conventions, style guides or regulations relating to usability. *Formula:* $X = \frac{1-A}{B}$ *A* denotes the number of usability compliance items that have not been implemented during testing and *B* the total number of usability compliance items specified. For $0 \le X \le 1$ holds: the closer *X* is to 1.0, the better the result. *Threshold:* Table 5 *ISO/IEC 25010 Category:* User interface aesthetics	44
111	Weight Methods for Class (WMC)	This metric corresponds to the number of all complexities of all methods of a class. *Formula:* $\text{WMC} = \sum_{i=1}^{n} c_I$ *Threshold:* Table 5	35
112	Worst Case Execution Time Analysis (WCET)	*Description:* This metric is an indicator estimation determines the actual worst case based upon the facts derived in the earlier phases.	78